PCC

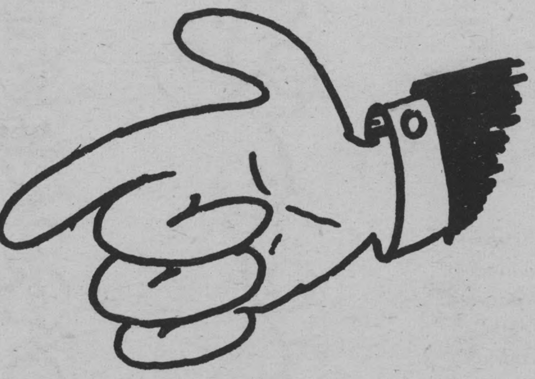
PEOPLE'S COMPUTER COMPANY is published 5 times during the school year. Subscriptions begin with the first issue, October 1972.

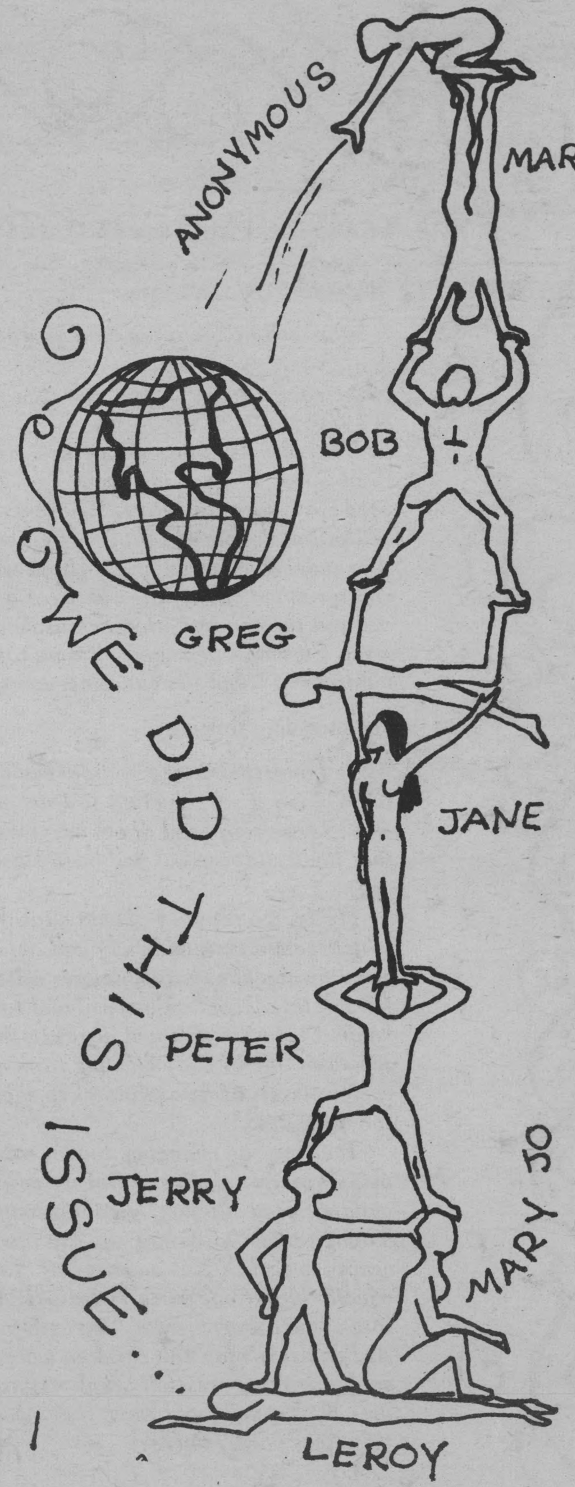
Single subscriptions — $4 for 5 issues [$5 Canada and overseas]

Group subscriptions, mailed all to the same address —

10 or more	*$3.00 each*
30 or more	*$2.50 each*
100 or more	*$2.00 each*

Subscription coupon on back cover





Cover art by Marie Marcks reprinted courtesy of Kaiser NEWS © 1967

!!!!! FOR PCC SUBSCRIBERS ONLY !!!!!

FOR A MERE PITTANCE, WE WILL SEND YOU TAPES OF GAME-PLAYING OR OTHER PROGRAMS THAT HAVE OR WILL APPEAR IN PCC. HERE ARE THE PRICES.

PROGRAM	*ISSUE + PAGE*	*PRICE*
NUMBER LETTER	OCT 72, PG 8 OCT 72, PG 11	$1 FOR BOTH
STARS	DEC 72, PG 3	$1
MELODY	DEC 72, PG 17	$1
TRAP	FEB 73, PG 8	$1
CHOMP	FEB 73, PG 9	$1
MUGWUMP	APR 73, PG 3	$1
HURKLE	APR 73, PG 22	$1
REVERSE	NEXT ISSUE	$1

```
***                   ***
* MINIMU~~N~~M ORDER $2.00 *
***                   ***
```

well, nobody's perfect

!!!!! FOR PCC SUBSCRIBERS ONLY !!!!!

WORKSHOPS IN COMPUTER SCIENCE

Instructors: ROBERT ALBRECHT, People's Computer Company and LEROY FINKEL, Ravenswood High School

Schedule: March 31—April 1; 9 a.m. – 10 p.m. Saturday; 9 a.m. – 6:30 p.m. Sunday; Lawrence Hall of Science, Berkeley campus

Credit: Two quarter units in Computer Science, each course

Fee: $65 each course, includes laboratory fee and some materials. Enrollment is limited

For information telephone 642-1061 in Berkeley

Computers in the Classroom X 402A (2)

An intensive "hands on" introduction to the use of calculators and computers in elementary and secondary school education. Participants use programmable calculators and learn the programming language BASIC, using both timesharing terminals and small computers. Methods for effectively using computers in the classroom and analysis of available hardware, instructional materials, computer education programs, and sources of further information. The course is conducted as an open classroom with activity centers for mathematics, science, business education, and social science teachers. It spans all grade levels — elementary through college. No previous programming or data processing knowledge is necessary.

Computers in the Classroom: Individualized Instruction X 402B (2)

This course is a continuation of Computers in the Classroom X 402A, and it gives participants the opportunity to increase their computer problem-solving skills. The course is run concurrently with X 402A, which is a prerequisite.

Games Computers Play X 407 (2)

Spend a weekend matching wits with a computer. Participants play computer games and explore both real life and "worlds of if" through the medium of computer simulation. Games of skill, games of chance, and games to learn by. Computing equipment is available throughout the course. No previous computer experience is required. The course is run concurrently with Computers in the Classroom X 402A and B.

LEARNING FAIRS and FUTURE FESTIVALS

The story starts like this:

*It happened at Peninsula School, a forty-seven-year-old family-staff-owned cooperative, the "oldest free school in the country," sit*e *of the first New Schools Conference in 1969.*

What was it? Fifteen "workshops and playshops" – Bead Game Music, Kids Teaching Kids, GestaltSmorgasbord . . . – and forty ongoing events – Mobile Solar Sculpture, People's Computer Center, Tree Loom, the Alexander Technique . . . . All spread over two days and sprinkled among the oak trees of the six-acre school site and the rooms within Peninsula's main building, a green Victorian mansion. "Come together as participators/innovators," read the announcement.

and ends like this:

While I am still cutting up bookshelves, the fair comes to its close. I am reluctant to leave it, and I realize why: I have reopened doors into ways of learning that I had shut or that had been closed upon me years ago.

For I am a product of that educational process by which we are systematically and deliberately weaned away from what Jerome Bruner called the "left-handed" (visual, intuitive, imaginative) and toward the "right-handed" (verbal, rational, logical), that process which separates one kind of learning from another and rates each in terms of its usefulness to society and not to the individual.

That may be changing now in many schools for many younger children. But we adults and older children need support, encouragement, what Barney Young called "loosening up," to open the doors again–to realize that a variety of "life games" are equally important parts of our education. And that sort of encouragement a "learning fair" can provide. At Peninsula's fair the children were there as our guides, making creativity look natural and easy as pie. We learned from them that weekend. But for me it was only a start.

from "Learning Fair" by Susan Sands, *Saturday Review of Education*, January 1973, pages 37–41.

Saturday Review of Education, Box 2043
Rock Island, Illinois 61207

If you missed the *Super Ecstatic Completely Credible Learning Fair* at Peninsula School last October – you can still make one of these this spring.

TO TOUCH TOMORROW: FUTURE FARE

"The merit of the future is that it is the area in which we can exert our will."

Bertrand de Jouvenal

If you feel with de Jouvenal that we don't have to accept the future as "fixed," then join us on March 30–31 and experience with us some of the exciting (and frightening) developments which are shaping our futures. Experience with us also the possibility of creating alternative futures to "the future" that you might feel is being imposed on you.

We begin on Friday afternoon, March 30, with a series of films to be shown in Olney Hall. Among the films you will see are:

"The World of Future Shock: Crisis in the 800th Lifetime" in which Alvin Tofler, author of Future Shock, examines the concept of "future shock" and the stress placed on individuals by a society in constant flux.

"The Family of the Future" looks at 3 different family lifestyles today which may be typical styles for tomorrow. The film is narrated by Margaret Mead.

There will be many others. These films will also be shown Saturday.

Friday evening at 8:00 PM in Olney Hall, Arthur C. Clarke, author of Profiles of the Future *and many other books about futures, will speak on "The Year 2001 and Beyond."*

Saturday, March 31, is Future Fare Day. Our environment will be Harlan Center and its adjacent outside spaces. Come play with a computer from the People's Computer Co.; imagine with Aaron Hillman you're Lost in Space; participate in The Future State of the Nation with Paul Twelker and Ken Layden; join Gloria Loventhal and her elementary school children in their "School 2000"; build a dome with Toni Ricci; eat with the One World Family Commune; ponder the prospects of Immortality with Chad Everone. These are just a sampling of the exciting events in store for you. So come, Touch Tomorrow.

for more info, contact

Ms. Sydney Goldstein
Director of Public Events
College of Marin
Kentfield, CA. 94904
(415)454-3962

ALTERNATIVES LEARNING FESTIVAL
A CELEBRATION

We at Webster College, in conjunction with the alternative schools in St. Louis, will be sponsoring a National Festival on Alternatives in Learning, to be held in St. Louis on May 3–6. Our hopes for holding such a festival are many; however, our specific aims are to learn more about ways we can humanize the diverse educational needs of an ever-changing and increasingly complex society.

The scope of ALF will be between 5,000–10,000 people from all over the nation. Some of the better known speakers so far are: Swight Allen, University of Massachusetts School of Education; Nate Blackman, Principal of Chicago Metro Alternative High School; Don Glines, author of **Creating Humane Schools;** *Joh Kozol, author of* **Free Schools;** *and Don Moore, Midwest Center for New Schools.*

ALF will revolve around a "Learning Bazaar," to be run by teachers, administrators, parents, students, and others involved in the creative learning experience.

Booths will focus on individualized experiencing . . .

Workshops will focus on group experience - - doing, thinking, creating . . .

Anyone interested in presenting a workshop, group session, learning shop, or other alternative involvement, at the FESTIVAL, or anyone desiring additional info, please call Webster College, (314)968-0500, ext. 400.

Webster College
470 East Lockwood
St. Louis, Mo. 63119

PAGE 2

marc's GRIPE

Recently I was modifying a program we had on an HP 2000F system. A simplified version of the program would look like this:

```
100 REM 1
150 REM 2
300 REM 3
350 REM 4
400 REM 5
450 REM 6
500 REM 7
550 END
```

I wanted to move lines 400 and 450 to before Line 300 so that the program would look like this:

```
100 REM 1
150 REM 2
200 REM 5
250 REM 6
300 REM 3
350 REM 4
500 REM 7
550 END
```

So I innocently typed

```
REN - 200, 50, 400, 450
```

(i.e., "Renumber Lines 400 through 450 by 50's, starting at Line 200" for those of you who are unfamiliar with HP BASIC.)

And the computer typed back

```
RENUMBER SEQUENCE OVERFLOW/OVERLAP
```

(Whatever that means) I assumed that the computer was refusing to perform this trivial operation and proceeded to do it the hard way. Here's how it looks (assume the program's name is PROG).

```
GET-PROG
NAM-T1
DEL-300,550
SAV
GET-PROG
NAM-T2
DEL-100,350
DEL-500,550
REN-200,50,400,450
SAV
GET-PROG
NAM-T3
DEL-100,150
DEL-400,450
SAV
SCR
NAM-PROG
APP-T1
APP-T2
APP-T3
KIL-T1
KIL-T2
KIL-T3
KIL-PROG
SAV
```

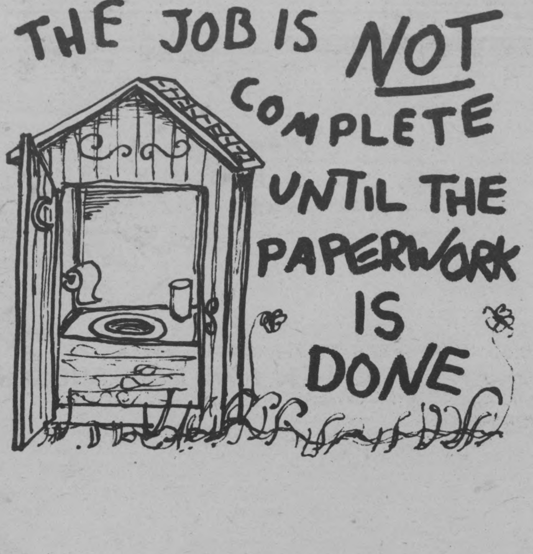
Come on HP, that's 27 lines I had to type instead of spending my time on something useful. Not only that, it could have been done with just one command, if HP's software was written properly to begin with.

Some day computer manufacturers are going to learn that computers are for people to use, and not the other way around.

That day will come only when computer users take it upon themselves to pull themselves out of their apathetic morass of ignorance of what computers *could* be doing for them, and *demand* their rights as consumers. If you read the label on a 25¢ can of soup why not read the "label" on a $100,000.00 computer system

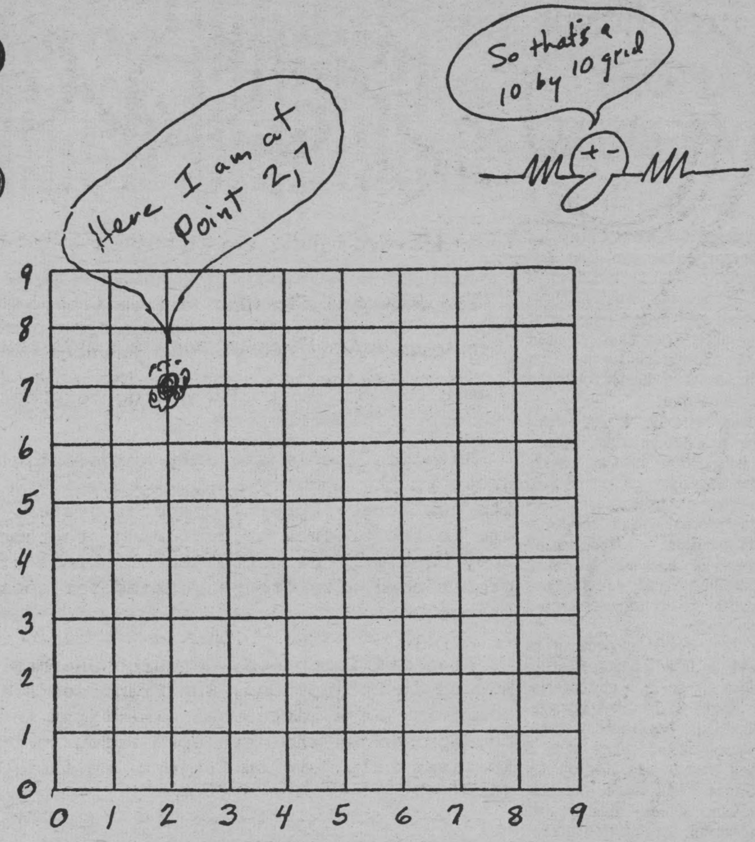
P.S. It matters not to the poor user if there is some undocumented "secret" way to do what he or she want.

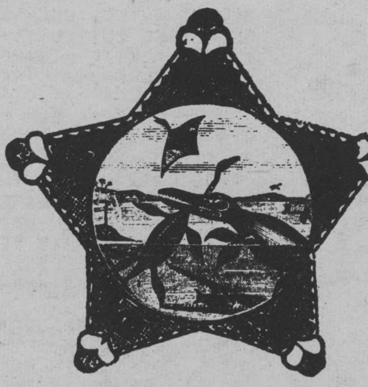
A graffiti found at Resource 1 is quite lucid on this point:

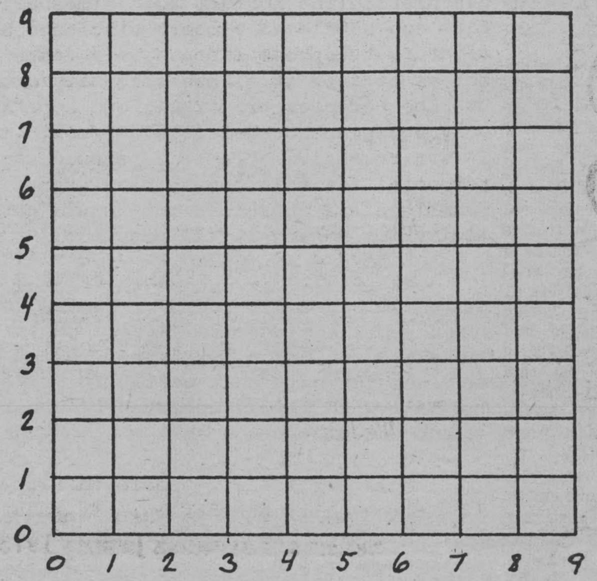


MUGWUMP

```
100  REM *** MUGWUMP - A HIDE AND SEEK GAME
110  REM *** PEØPLE'S CØMPUTER CØMPANY, MENLØ PARK CA
120  RANDØM
130  REM *** G=GRID SIZE  N=NUMBER ØF GUESSES ALLØWE
140  LET G=10
150  LET N=5
160  PRINT "DØ YØU WANT THE RULES (1=YES  0=NØ)";
170  INPUT Z
180  IF Z <> 1 THEN 350
190  REM *** RULES IN LINES 200 THRU 330
200  PRINT "A MUGWUMP IS HIDING IN A";G;"BY";G;"GRID. TRY TØ"
210  PRINT "FIND HIM BY GUESSING HIS GRIDPØINT. HØMEBASE IS"
220  PRINT "GRIDPØINT  0,0  AND A GUESS IS A PAIR ØF WHØLE"
230  PRINT "NUMBERS (0 TØ";G-1;") SEPARATED BY A CØMMA. THE FIRST"
240  PRINT "NUMBER IS THE DISTANCE TØ THE RIGHT ØF HØMEBASE"
250  PRINT "AND THE SECØND NUMBER IS THE DISTANCE ABØVE THE"
260  PRINT "HØMEBASE. FØR EXAMPLE, IF YØU THINK THE MUGWUMP"
270  PRINT "IS HIDING 8 UNITS TØ THE RIGHT ØF HØMEBASE AND"
280  PRINT "3 UNITS ABØVE HØMEBASE, THEN ENTER  8,3  AS YØUR"
290  PRINT "GUESS AND PRESS THE 'RETURN' KEY."
300  PRINT
310  PRINT "YØU GET";N;"GUESSES. AFTER EACH GUESS, I WILL"
320  PRINT "TELL YØU HØW FAR (IN A DIRECT LINE) YØU ARE FRØM"
330  PRINT "THE MUGWUMP. "
340  REM *** HIDE MUGWUMP AT RANDØM GRIDPØINT  A,B
350  LET A=INT(G*RND(0))
360  LET B=INT(G*RND(0))
370  PRINT
380  PRINT "MUGWUMP IS HIDING. YØU GET";N;"GUESSES."
390  REM *** N GUESSES ALLØWED - LINES 400 THRU 560
400  FØR T=1 TØ N
410  PRINT
420  PRINT "WHAT IS YØUR GUESS";
430  INPUT X,Y
440  REM *** IF MUGWUMP NØT FØUND GØ TØ LINE 500
450  IF X <> A THEN 520
460  IF Y <> B THEN 520
470  PRINT "YØU FØUND HIM IN";T;"GUESSES!!!"
480  PRINT "LET'S PLAY AGAIN."
490  PRINT
500  GØTØ 350
510  REM *** D=STRAIGHTLINE DISTANCE TØ MUGWUMP
520  LET D=SQR((X-A)↑2+(Y-B)↑2)
530  REM *** THEN WE RØUND D TØ ØNE DECIMAL PLACE
540  LET D=INT(10*D)/10
550  PRINT "YØU ARE";D;"UNITS FRØM THE MUGWUMP."
560  NEXT T
570  REM *** MUGWUMP NØT FØUND IN N GUESSES
580  PRINT
590  PRINT "SØRRY, THAT'S";N;"TRIES."
600  PRINT "MUGWUMP IS AT GRIDPØINT ";A;",";B
610  PRINT "LET'S PLAY AGAIN."
620  PRINT
630  GØTØ 350
640  END
```

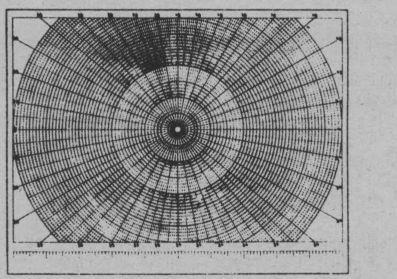


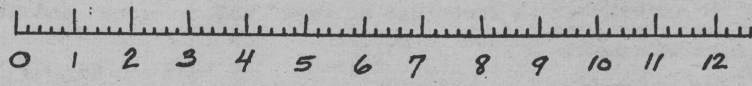




You want

- Larger grid... smaller grid? Change grid size G in Line 140
- More guesses... fewer guesses? Change number of guesses N in Line 150.





You can cut this out and use it as measuring stick on the gri

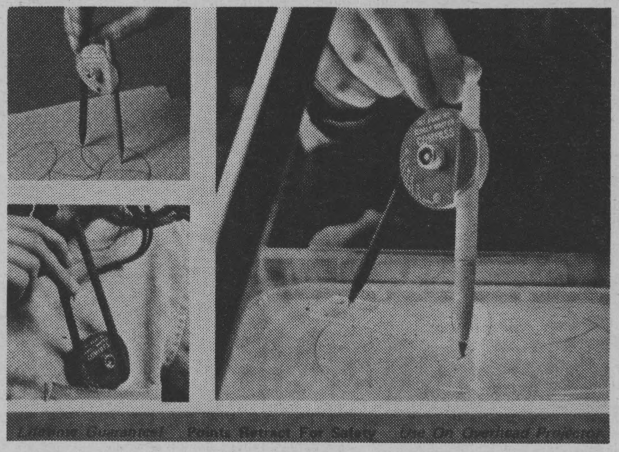
MUGWUMP was inspired by Project SOLO Module #0201.
Contact Project SOLO, Computer Science Department, University of Pittsburgh 15213.

TOOLS TO HELP YOU FIND MUGWUMP

Buy them from

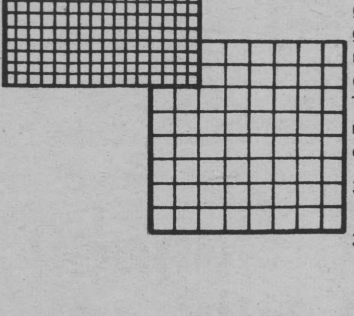
CREATIVE PUBLICATIONS
P. O. BOX 10328
PALO ALTO, CALIFORNIA 94303
Business Office – (415) 968-3977
Product Development and Teaching Center – (415) 968-1101

... or make them yourself...


Lifetime Guaranteed   Points Retract For Safety   Use On Overhead Projector

CIRCLE MASTER COMPASS

34100	**Circle Master Compass—Individual . . .**	**$ .90**



TRANSPARENT GRIDS (INCH & CM)

Flexible, transparent acetate grids for many different purposes. Measure area by placing grid over plane figure. Put blocks on top of grid. Demonstrate fractions, decimals, area relationships on the overhead projector. Make charts, graphs, etc.—then wipe them clean! Two kinds of grids available—10" x 10" ruled in inches or 25 cm x 25 cm ruled in centimeters.

36500	Transparent Cm Grids – (Pkg. of 10)	$3.25
37000	Transparent Inch Grids – (Pkg. of 10)	$3.25

36900 NEW MATH RULE–INDIVIDUAL . . . . $ .35

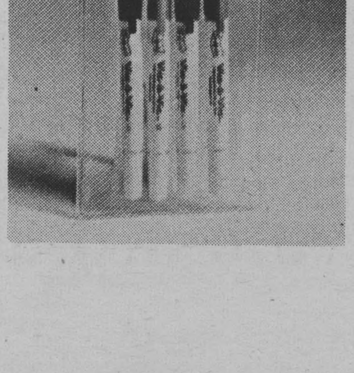
Durable four-in-one ruler made of plastic. Has four scales – two on each side – 1/10", 1/16", metric and number line. Accurate, easy-to-read calibrations.

VIS–A–VIS PENS

Visual aid pens for overhead projector transparencies which write smoothly and evenly on all acetates. Bright, transparent colors do not change during use. Easy to clean from acetates; washable from hands and clothing; non-toxic and odorless. Available in Pocket Set (one each of the four colors – black, red, blue and green) or in boxes of a dozen in black only.

38450 VIS-A-VIS PENS

Pocket Set (4)	**$2.40**
Dozen black	**$6.00**



BA

San Francisco Chronicle

Sat., Jan. 8, 1972

Free-Forms At 'One' Warehouse

By Thomas Albright

Except for its hot mustard exterior paint job, the immense, six-level building at the corner of Tenth and Howard streets looks like any of the older warehouses and light industrial plants that form most of the surrounding neighborhood.

Once inside, however—via a door marked simply, "One" — you plunge into an utterly mind-boggling complex of winding corridors and free-form rooms, and of free-form personalities, specialized know-how and sophisticated technology that are coming together in an unprecedented new way.

I had come expecting to meet a few artists sharing studio space in a charateristic South of Market garage-loft. "One" contains a sizeable amount of space that is being used as studios by painters, sculptors, ceramists and other artists and craftsmen. But it also houses fully equipped television and radio studios; film, photo, and video processing labs; a computerized data bank, experimental free school and professionally-manned clinic; and a staggering variety of other, continually evolving and overlapping activities.

ACTIVITIES

These activities are the work of some 200 people and 60 organizations who are involved in "One" on a regular basis, and who range from students and recent university graduates and drop-outs, to PhDs, electronics specialists and other highly trained professionals who have pulled out of the normal stream of economic life to join in a cooperative sharing of talent, experience and physical resources. They bring with them not only knowledge and skill, but often costly and highly specialized technological hardware as well.

A year and a half old, "One" is the senior member of a loose network of similar warehouse "projects" that has already grown to include a larger "Artaud," at 17th and Alabama streets, and several smaller complexes. In the works are others in San Francisco and the East Bay.

The idea for "One" originated with Ralph Scott, an architect and engineer by training who was then sharing space with a dozen other people in smaller warehouse quarters.

At its most practical level, Scott explained, the plan grew out of the fact that there were "a lot of people who needed to find an alter native way to deal with eco nomic problems, such as low income, or unwillingness to sacrifice their integrity for a reasonable salary. There are great numbers of poeple today who have talent, experience and a command of money, but refuse to pay the price. There are others — people with master's degrees and PhDs — who simply can't find jobs."

ISOLATION

"A second factor is isolation, which makes poeple living in a city relatively ineffective. People live in a highly compartmentalized world, with no flow or continuity between one activity and another. Many creative people are victimized by the negative aspects of living in an urban environment and are unable to take advantage of its positive elements, such as stimulation and sharing ideas. Some can't pay to keep a phone installed to call across the city and find out what a friend is doing."

Finally, Scott pointed out, "there is an aura of elitism, and even mysticism, that surrounds the world of technology, and many of the arts, because so many people lack access to equipment, training and other resources."

Thus, "One" took root from the need to "put it together" in the shape of an organically inter-related and evolving community that would embrace a wide range of skills and resources, with as few physical or temporal barriers as possible between creativity, work, life and education.

PROPOSAL

Scott formulated a proposal, an associate, Craig Brown, went on the radio to talk about it, and within three days a dozen people were manning a switchboard to receive calls from individuals and collectives interested in joining.

A series of general meetings was convened, and eventually the group, by then numbering about 100, was able to negotiate a five-year $50,000 per annum lease on the 84,000 square foot building — a one time candy factory which had long stood vacant — from, as Scott puts it, "a very supportive landlord."

The building that "One" took over was a largely empty shell of bare cement exterior walls, floors and large pillars supporting the ceilings that divide its six levels — a basement, four storeys and a smaller, roof-top "penthouse."

TASK

The group's first task was therefore an extensive renovation job, dividing the building into "spaces" (the word "room," with its connotation of compartmentalization, is avoided at "One") appropriate to the needs of their occupants.

In keeping with "One" philosophy, its architects and trained builders largely stood aside while partitions — sheetrocking, firedoors — were designed and erected by occupants inexperienced in any kind of construction work, after a "class" in wall-building conducted by a contractor friend of Scott's.

"This work is all up to code," he pointed out. "In the process, everyone learned new skills — wiring painting, plumbing. People spend an enormous lot of time everyday in the midst of things they have no understanding of, even though some are essential to survival. Here everyone learns the fundamentals connected with the building, and shares in its maintenance."

RENT

Occupants pay 6½ cents per square foot for their spaces, which are generally measured in terms of "bays," or the area between four of the large interior pillars. A "bay," Scott points out, is equivalent to an average three-room apartment, and rents for $23 a month. Many spaces are considerably larger than a single bay,

- Fort Help. Established over a year ago by the psychiatrist, Dr. Joel Fort, the drop-in clinic offers consultation on a broad range of drug, sex and other special medical and psychological problems, under a staff of trained professionals helped by numerous volunteers. It also administers the City's only private methadone program.
- D.B. Associates, a full-fledged manufacturing plant of electronics parts and equipment which also serves as a school for the "demystification" of electronics technology.
- An experimental free school for high-school age students. Presided over by a pair of Ph.D. graduates from the Union Graduate School, the school itself is a big room filled with old, overstuffed furniture and lined with shelves of books. Its "classrooms," however, also include D. B. Associates, Airwaves, and everything else that takes place inside the building.
- A School of Holography. Conducted by the noted holographer, Lloyd Cross, the school contains four individual studios equipped for laser photography and projection of three-dimensional images.
- Ecos. This organization includes a pool of professional talent ranging from structural engineers to ecologists who deal with problems that "One" and its members encounter, and share with them their knowledge of the intricate workings of the law, City Government and similarly "occult" forces.

Ecos also makes information and aid available to other groups who are interested in setting up warehouse projects. Among its prominent features is a huge wall map of the City which pinpoints the locations of some 60 vacant warehouses that may someday be available for occupancy by the hundreds of persons coming to "One" to look for space.

Resource One

Reprinted from WORKFORCE (January and February, 1973), published by Vocations for Social Change, Box 13, Canyon, CA. 94516. VSC is a tax-exempt, anti-profit collective subsisting entirely on your donations.

Pam Hardt and Bernard Greening for the Resource One Collective.

Resource One is the only grassroots community group in the country that we know of to obtain its own large-capacity computer system. They are in the process of designing programs that they hope will be useful to all kinds of non-profit community groups working for social change.

From its inception, Resource One has been housed in Project One, San Francisco's first community in a warehouse. The first task of the four of us who developed Resource One was to first help develop Project One from June '70-June '71.

In midsummer, 1971, Transamerica Computer Co. was just a number in the telephone book. A man we reached in their public relations dept. said TA had "a warehouse full of computers," which turned out to mean three Xerox Data Systems 940 computers returned by their lease customers. (The XDS-940 is a timesharing computer that can handle many users simultaneously over separate telephone lines. This feature is ideal for us because it allows community groups to use the computer from their own location. The only equipment needed is a $45/month terminal.) TA was receptive because we showed them the potential for relating to many groups in San Francisco and therefore they could get good publicity for making it all possible.

HOW

Very
the
editing
here
please!

TA was also having trouble getting rid of their 940's because they originally cost $800,000 seven years ago and are "second generation." Customers could now buy faster "third generation" machines for half that cost. In September 1970, the deal with TA was approved by their top management.

In November 1971, we recieved $10,000 from the Stern Family Foundation. It wasn't easy to get. Resource One had a corporate identity (we took over the tax-exempt corporation from the old San Francisco Switchboard) and a promised computer, but no operating track-record. We were able to use the efforts of the Ecos Project--which was also helping other warehouse communities develop--and the demonstrable credibility of the 150 people building; a living-working environment for themselves at Project One to show that we were really serious and had a chance to succeed. Ecos initiated the Stern contact, and convinced them that technology is an integral part of any new, viable urban environment. Stern granted a total of $25,000, which we split; Ecos taking $15,000 and Resource One $10,000.

W

The money from Stern allowed us to begin the design and construction of the machine room. The computer needed a dust free, air-conditioned environment. We wanted it to be a visually accessible yet flexible shell. With the help of two architects in Ecos, we finally drew up plans for a sheetrock structure with Lexan (a fire-retardant plastic) windows.

Toward the end of January, 1972, the machine room was under way and we were down to $4,000 which had to be spent soon. We had been contacting and building relationships with foundations for months. We only received responses from foundations with whom we managed face-to-face contact with the director. So far, only those who have also seen us at work have come through. A problem has been that the foundations we encountered did not have the technical expertise to evaluate our project; they seemed threatened by the quantum jump from simple telephone switchboards to computerized information networks. However, we did convince the people of the Whole Earth Catalogue Community Fund to lend us $8,000 "for a month" to tide us over until the other grants came in.

By May, we had used most of the WEC money, when two foundations each announced grants of $25,000 contingent on our raising a total of $100,000. We had not expected the contingency part, but resolved to use it as a lever in

INTERRUPT ARTICLE

INTERRUPT

newsletter of computer people for peace

AT A CPP MEETING IN NEW YORK LAST MARCH (REPORTED IN INTERRUPT 15) THE TOPIC ''THE POSITIVE USES OF COMPUTERS--ARE THERE ANY\'' WAS DISCUSSED. THE PREVAILING OPINION SEEMED TO BE WELL, MAYBE SO, POTENTIALLY, BUT NOT YET. RESOURCE ONE IS DOING ALL IT CAN TO CREATE REALITY FROM WHATEVER POTENTIAL THERE IS.

RESOURCE ONE IS A COLLECTIVE OF PEOPLE FROM DISPARATE BACKGROUNDS, WHO BELIEVE THAT TECHNOLOGICAL TOOLS CAN BE TOOLS OF SOCIAL CHANGE WHEN CONTROLLED BY THE PEOPLE. WE'RE TRYING TO HELP THEM BECOME AVAILABLE TO ALL PEOPLE, AND TO ENCOURAGE AND ASSIST THE DEVELOPMENT OF WAYS THESE TOOLS CAN IMPROVE ALL OUR LIVES.

OUR PRINCIPAL TOOL AT PRESENT IS AN XDS-940 COMPUTER, A MEDIUM SCALE, SECOND-GENERATION TIMESHARING MACHINE WHOSE CAPABILITIES SEEM WELL SUITED TO OUR PURPOSES BECAUSE OF THE 940'S ABILITY TO INTERACT WITH A LARGE NUMBER OF INDIVIDUAL OR GROUP USERS SIMULTANEOUSLY (OVER TELEPHONE LINES) AND INEXPENSIVELY.

SOME OF OUR PRIMARY PROJECTS ARE THE BUILDING OF A RETRIEVAL SYSTEM GEARED TO THE NEEDS OF NON-ESTABLISHMENT REFERRAL SERVICES (SWITCHBOARDS, HOTLINES, ETC.) ENABLING THEM TO CREATE AND SHARE LARGE DATA BASES, THE ASSISTANCE OF VARIOUS GRASSROOTS POLITICAL AND ECONOMIC RESEARCH PROJECTS AIMED AT COMMUNITY ORGANIZING, THE CHEAP PRODUCTION OF MAILING LISTS, A COMMUNITY MEDICAL CLINIC INFORMATION SYSTEM, A PROGRAM OF COMPUTER EDUCATION AND DEMYSTIFICATION FOR THE GENERAL PUBLIC, AND GENERALLY RAISING THE LEVEL OF COMMUNICATION AND COOPERATION AMONG DIFFERENT COMMUNITY GROUPS.

RESOURCE ONE'S ROOTS GO BACK TO BERKELEY DURING THE CAMBODIA CRISIS OF MAY, 1970. A GROUP OF COMPUTER PEOPLE THERE GOT TOGETHER, LIKE OTHERS, AND TALKED OF THEIR DISENCHANTMENT WITH HOW THEIR SKILLS WERE DESTINED FOR USE IN THE SYSTEM. THEY FANTASIZED ABOUT USING COMPUTERS FOR BUILDING COMMUNICATION NETWORKS, AND SEVERAL MONTHS LATER WERE ATTRACTED TO PROJECT ONE IN SAN FRANCISCO, WHERE OTHER TECHNOLOGICALLY-ORIENTED PEOPLE, AS WELL AS ARTISTS AND EX-PROFESSIONALS OF ALL TYPES, WERE GATHERING TO TRY OUT A NEW CONCEPT OF INTEGRATING THEIR SKILLS AND WORK WITH THE REST OF THEIR LIVES.

PROJECT ONE (OR ''ONE'') WAS A VACANT 5-STORY WAREHOUSE BUILDING IN DOWNTOWN SAN FRANCISCO--84,000 SQUARE FEET OF BARE, COLD CONCRETE, WHICH HAS SINCE BEEN TRANSFORMED INTO AN IMAGINATIVE WARREN OF ''SPACES'' IN WHICH 90 PEOPLE LIVE AND 150 PEOPLE WORK ON A WIDE VARIETY OF PROJECTS. BESIDES RESOURCE ONE, THERE ARE AN EXPERIMENTAL HIGH SCHOOL, A VIDEOTAPE-PRODUCING GROUP, MUSIC AND RADIO RECORDING/PRACTICE STUDIOS, A FILM PROCESSING LAB, THEATER REHEARSAL SPACE, AND OFFICE AND/OR LIVING SPACE FOR THE NORTHERN CALIFORNIA H.Q. OF VVAW, A RADICAL WELFARE DEPARTMENT WORKERS' UNION, A COUNSELING CENTER, AND NUMEROUS ART AND CRAFT PEOPLE.

''ONE'S'' SOCIAL AND PHYSICAL ENVIRONMENT WAS BUILT COMPLETELY BY ITS MEMBERS, TO THEIR OWN SPECIFICATIONS, AND IN THE PROCESS THEY SHARED THEIR TOOLS, SKILLS AND POLITICAL IDEAS AS ONLY PEOPLE WHO MUST DEPEND ON EACH OTHER CAN. A WORKERS' COLLECTIVE HAS BEEN FORMED TO FURTHER DEVELOP AND PROFIT FROM NEWLY-ACQUIRED CONSTRUCTION SKILLS. THE COMMUNITY IS ''RUN'' BY UNANIMOUS CONSENSUS; DECISIONS ARE MADE AT WEEKLY MEETINGS.

THE FIRST YEAR OF ''ONE'' WAS A WEEDING-OUT PROCESS WHICH LEFT ONLY PAM HARDT FROM THE ORIGINAL BERKELEY GROUP. PAM WAS CALLING COMPUTER MANUFACTURERS ON THE PHONE TRYING TO LAND A DONATION OF A MINI-COMPUTER, WHICH COULD BE TAKEN AROUND COMMUNITIES FOR DEMONSTRATIONS AND EDUCATIONAL EXPOSURE. TRANSAMERICA COMPUTER CO. (UNDER ''T'') WAS THE FIRST BITE. TA HAD 3 940'S SITTING UNUSED IN A WAREHOUSE, RETURNED FROM LEASE BECAUSE THEY HAD BEEN REPLACED BY NEWER MACHINES. ALTHOUGH THE 940 IS STILL BEING USED BY COMMERCIAL TIMESHARING SERVICE COMPANIES, THERE ARE FASTER, CHEAPER 3RD GENERATION MACHINES AVAILABLE WHICH MAKES THE 940 UNATTRACTIVE FOR NEW INSTALLATIONS, AND THEREFORE ''OBSOLETE'' COMMERCIALLY. TA RESPONDED TO THE IDEA OF USING A COMPUTER DONATED BY THEM FOR AN INFORMATION/REFERRAL NETWORK THAT WOULD TIE ALTERNATIVE GROUPS (SWITCHBOARDS) INTO ESTABLISHED SOCIAL SERVICE AGENCIES (INFORMATIONALLY), WITH GOOD PUBLICITY POTENTIAL FOR TA.

SO, IN SEPTEMBER 1971 WE HAD A COMPUTER, BUT NO MONEY, NO SOFTWARE, NO I/O EQUIPMENT, NO PLACE TO INSTALL THE COMPUTER, NO DEVELOPED ''PROGRAM'' OF WHAT TO DO, AND VERY FEW COMMITTED PEOPLE. WE DID, HOWEVER, HAVE THE SUPPORT OF OTHER MEMBERS OF ''ONE''--IN PARTICULAR THE ECOS PROJECT, WITH WHOM WE PRESENTED A JOINT FUNDING PROPOSAL TO THE STERN FOUNDATION WHICH NETTED $10,000. USING THIS MONEY TO BUILD THE COMPUTER ENVIRONMENT ($5,000 WORTH), PAY RENT AND TELEPHONE BILLS (NO SALARIES), WE DEVELOPED A PROPOSAL FOR A FULL-SCALE COMMUNITY COMPUTER CENTER WITH A BUDGET OF $100,000 FOR ITS FIRST YEAR. IT TOOK US A YEAR OF STEADY FUNDRAISING TO RAISE THE MONEY, DURING WHICH TIME THE COMPUTER WAS INSTALLED (AT A TOTAL COST OF $700), AND WE BEGAN TO LEARN ABOUT OUR OPERATING SYSTEM, WHICH NONE OF US HAD SEEN BEFORE.

PETER DEUTSCH BROUGHT UP THE OPERATING SYSTEM (WHICH HE HAD WRITTEN 3 YEARS BEFORE), DONATING HIS TIME, AND FRED WRIGHT FROM THE STANFORD ARTIFICIAL INTELLIGENCE PROJECT HELPED OUR RESIDENT HARDWARE PERSON, LEE FELSENSTEIN, GET EVERYTHING WORKING. THE LAST MAJOR CONTEMPLATED SYSTEMS PROJECT--A RETRIEVAL SYSTEM--IS NOW BEING IMPLEMENTED BY BART BERGER, BERNARD GREENING AND JOHN COONEY OF OUR STAFF, WITH THE HELP OF ROBERT SHAPIRO OF META INFORMATION APPLICATIONS IN CAMBRIDGE (HE DESIGNED IT FOR THE PDP-10). PAUL HECKEL, WHO WORKS WITH DEUTSCH AT XEROX PARC, HAS HELPED US GENERALLY TO ACQUIRE NEEDED HARDWARE AND SOFTWARE.

BY APRIL, ALL OF OUR SYSTEMS (INCLUDING ONE TO PROVIDE STATISTICAL REPORTS TO OUTPATIENT CLINICS) WILL BE FULLY OPERATIONAL, AND WE WILL HAVE BEGUN ACCEPTING INPUT OF SEVERAL LARGE DATA BASES, INCLUDING REFER INFORMATION FOR SWITCHBOARDS AND OTHER PEOPLE-ORIENTED MEDIA, DEMOGRAPHIC INFORMATION AND SUCH FOR RESEARCH PROJECTS, AND ''USEFUL CONTACT'' FILES FOR A HOPEFULLY LARGE NUMBER OF COMMUNITY GROUPS. THE ONLY REAL IMPEDIMENT TO INCLUDING SOME NATIONAL APPLICATIONS OF THE RETRIEVAL SYSTEM IS THE COST OF TELEPHONE CONTACT.

pursuing further grants. Fortunately, we received two contributions of $5,000 from individuals which enabled us to last until the fall, when the rest of our funding came through.

In everything else, we had sought the help of "experts" but wound up doing most of the actual work ourselves. We were, however, very lucky to find an experienced person in Los Angeles who brought up the machine in three days for a total cost of $600. Then Peter Deutsch, who wrote an operating system for the 940 several years ago and now does research for Xerox, put the "software" into operation by contributing his time for several weeks.

During the summer, the Resource One staff seemed to stabilize to one electronics engineer, 4 programmers, 2 electronic apprentices, and an accountant with business management experience. Internally, we had to fight people's tendency to define themselves by their acquired specialities and we had to develop some common vocabulary to be mutually understood. Dealing with people's professional conditioning is a day-to-day process which continues while we are developing our two major software systems; an information retrieval system and a medical statistical and reporting system.

The information retrieval system allows a data base, such as the community service information held by switchboards to be developed, easily updated, rearranged, and printed by categories. The system is designed to be used by the non-computer professional, who sees the computer as a tool in his/her work. And, it can alleviate the six month research time spent putting together essentially out-of-date directories, and can be used for several purposes, including power structure research, listing of referral information for groups which are in contact with people who need it, and other kinds of information storage. The system was designed by Robert Shapiro of Meta Information Applications. He is helping us implement the system, which should be completed by March, 1973.

The medical system will provide reporting and in-house statistical work for the O.E.O./H.E.W. funded health care centers. They have earmarked data processing funds and, generally, are not getting service comparable to the cost they are paying. Because we are local, we can provide more immediate reporting and are willing to tailor the system to suit health care center's particular needs. Selling this system can provide us with enough income to handle the statistical needs of the free clinic, which could not afford to buy data-processing services.

We will shortly offer a system for producing statistical tables describing a clinic's activities. Clinics will be able to use these reports to satisfy the requirements of their government grants. This system is intended to produce substantial revenues for Resource One--it may turn out to be our main bread and butter--but more important, it will allow us to develop other, perhaps more useful informational tools for the clinics which otherwise couldn't afford to pay for it.

In addition to the two soft-ware systems, we are developing a community education program. Using the computer and other media, such as video tape, we want to explore how computers are used in America, how their usage affects individuals, other possible uses of computers, and how computers integrate with other communication and transportation technology. Through our community education program, we hope to reach large numbers of people.

We are well under way towards making the computer we have available to community groups in the Bay Area. But, Resource One will continue to be seen as an odd or mysterious creation without broad implications unless technical people (architects, engineers, biologists, etc.) commit themselves to expanding and redefining the usage of their skills and the redistribution of control over technological hardware in conjunction with communities of people who are trying to generate change.

For more information CONTACT- Pam Hardt or
Bernard Greening, c/o Resource One, 1380
Howard St., San Francisco, CA 94103

AT THAT POINT TOO WE PLAN TO BEGIN GENERATING SOME MONEY WITH THE SYSTEM; OUR ONLY SIGNIFICANT REVENUES SO FAR HAVE COME FROM A TIMESHARING CONTRACT WITH THE UNIVERSITY OF CALIFORNIA. WE THINK IT'S IMPORTANT THAT EVERYONE DEVELOP A MEANS OF SUPPORT THAT DOESNT DEPEND ON THE LARGESSE OF FOUNDATIONS OR ON WELFARE. IT FOLLOWS THAT IF WE ARE TO BE VALUABLE TO OUR COMMUNITY, OTHER GROUPS WILL SHARE THEIR SUPPORT WITH US (ONLY TO THE EXTENT THEY ARE ABLE TO), AND THAT WE MUST ALSO FIND NON-EXPLOITATIVE WAYS TO MAKE MONEY IN THE MORE COMMERCIAL MARKET. THAT'S ONE REASON FOR THE MEDICAL INFORMATION SYSTEM, WHICH FORD TURPING AND CHRIS MACIE ARE FINISHING,
SINCE OUTPATIENT CLINICS OFTEN HAVE GOVERNMENT GRANTS THAT INCLUDE LINE ITEMS FOR DATA PROCESSING. AS A RULE, WE LOOK AT ANY USE OF THE 94C AS POTENTIAL REVENUE, UNLESS THE GROUP INVOLVED IS TOTALLY BROKE AND CAN'T EVEN COVER OUR MINIMAL OUT-OF-POCKET COSTS (WHICH IS SELDOM).

BESIDES KEEPING THE MACHINE UP, LEE FELSENSTEIN, MARK PEACOCK AND PAUL WARD ARE PUTTING TOGETHER AN ELECTRONICS SHOP TO BE USED BOTH FOR EDUCATION (THEIRS AND OTHERS!) AND FOR DESIGNING AND BUILDING HARDWARE NEEDED IN-HOUSE AND FOR OUTSIDE CONTRACTS.
ONE PROJECT WHICH HAS WIDER POTENTIAL IS A UNIQUE INTERCOM SYSTEM LEE DESIGNED FOR ''ONE'' THAT LINKS ALL THE SPACES IN THE BUILDING-- IT COULD LEAD TO MORE SOPHISTICATED NEIGHBORHOOD COMMUNICATIONS SYSTEMS.

OUR EDUCATIONAL PROJECTS HAVE BEEN THE MOST DIFFICULT TO CONCEPTUALIZE, BUT ARE PERHAPS THE MOST CRUCIAL TO EXPANDING THE CONCEPT OF HUMAN USES OF TECHNOLOGY. PEOPLE WE TALK TO SEEM MYSTIFIED AT FIRST EVEN WITH THE SIMPLE MAILING-LIST PROGRAM, AND WE ARE ALL CONSTANTLY EDUCATING NON-TECHNICAL PEOPLE, HOPING TO BREAK DOWN THEIR FEARS AND STIMULATE CREATIVE THINKING ABOUT HOW THEY CAN USE THE SYSTEM. STEVE ROBINSON, AN MBA, HAS BEEN GIVING BOOKKEEPING AND TAX CLASSES TO OTHER GROUPS, AND THERE IS SOME ''FALLOUT''--PEOPLE BECOMING EXPOSED WHEN THEY COME TO THE CLASS, AND LATER GETTING INTERESTED IN HAVING ACCESS TO THE COMPUTER. SIMILARLY, MIKE CHADWICK, THROUGH HIS SNOBOL CLASSES, AND PAM HARDT, THROUGH A SERIES OF VIDEOTAPES ON WOMEN'S PROBLEMS, ARE REACHING OUT TO THE COMMUNITY AND EXPOSING US IN THE PROCESS.

AS OF NOW, RESOURCE ONE'S FINANCES SEEM SECURE FOR A YEAR OR SO, AND NO REALLY DIFFICULT TECHNICAL PROBLEMS (BUT LOTS OF OPPORTUNITIES) REMAIN. YET WE ARE STILL STRUGGLING WITH THE REMNANTS OF OUR PROFESSIONAL ELITISM, NEEDS FOR SOME HIGHER AUTHORITY (THE BOSS) TO DEFINE OUR TASKS, AND TENDENCIES TOWARD SPECIALIZATION, ALL OF WHICH RESTRICT THE COLLECTIVE PROCESS WE WANT TO CULTIVATE. OUR DECISIONS ARE MADE AT RELATIVELY OPEN MEETINGS, BY UNANIMOUS CONSENSUS OF A SEVEN-PERSON STEERING COMMITTEE. FINAL RESPONSIBILITY FOR GETTING WORK DONE RESTS THERE. IT IS DIFFICULT TO BUILD A COLLECTIVE CONSCIOUSNESS, WHERE PEOPLE CAN SUPPORT EACH OTHER EMOTIONALLY AND OTHERWISE, WHEN WE ARE AT DIFFERENT LEVELS OF POLITICAL AWARENESS, WORK SKILLS, LENGTHS OF ASSOCIATION, AND COMMITMENTS TO THE CONCEPT OF COMMUNITY. BUT IT'S IMPORTANT, AND WE'RE TRYING HARD.

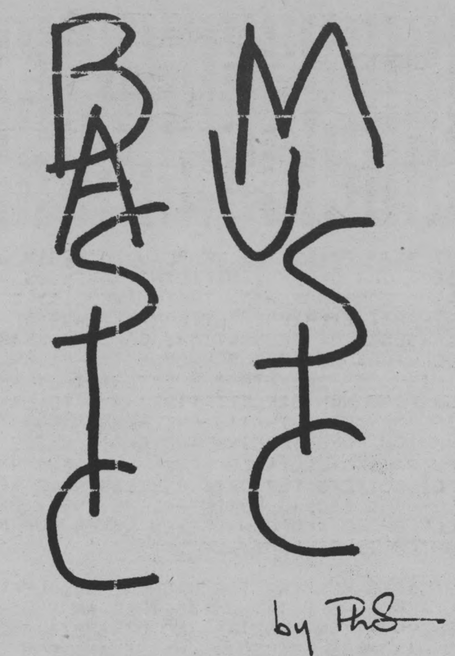
IT'S VERY CLEAR TO US THAT RESOURCE ONE COULD NOT HAVE GOTTEN OFF THE GROUND WITHOUT THE HIGHLY DEVELOPED SHARING ETHIC OF ''ONE'' -- AND WE CAN'T AFFORD TO FORGET THAT. WE WANT TO SHARE OUR EXPERIENCES WITH OTHERS INVOLVED IN SIMILAR PROCESSES, AND WELCOME ANY CONCRETE SUGGESTIONS ON POSSIBLE APPLICATIONS OF OUR TOOLS. SOMEWHERE, WE MISLAID THE ORGANIZATION MANUAL THAT SAYS WHAT A GROUP LIKE US IS SUPPOSED TO BE DOING, SO WE'RE PLAYING IT BY EAR.

THIS ARTICLE PREPARED BY:
FORD TURPING
LEE FELSENSTEIN
STEVE ROBINSON

Bob and othas

We feel the how is important as well as the "who" and "what'

In doing your piece could you enclude the text included in boxes verbatum or with very little editing



In this issue, we'll look at some practical music theory and describe a few "utility" algorithms for music programming.

The first article in this series described chromatic scales. The 12-tone chromatic scale is based on the 2nd overtone, which is twice the frequency of the fundamental or base frequency. The tones of the scale are generated by multiplying the fundamental frequency by the 12th root of 2, 12 times in succession (ending with the value 2).

Here is a general program for tempered (proportional) scales.

```
100 REM *** TEMPERED SCALE FREQUENCY GENERATOR ***
110 PRINT
120 PRINT "AT WHAT OVERTONE SHOULD I BEGIN REPEATING";
130 INPUT R
140 PRINT
150 PRINT "HOW MANY TONES IN THE SCALE";
160 INPUT T
170 PRINT
180 PRINT "WHAT IS THE BASE FREQUENCY";
190 INPUT F
195 PRINT
200 LET L=LOG(R)/T
210 PRINT "TONE","FREQUENCY"
220 PRINT
230 FOR I=0 TO T
240 PRINT I+1, F*EXP(I*L)
250 NEXT I
260 PRINT
270 END
```

If you input:

R = 2 (octave of the fundamental)
T = 12
F = anything

you obtain the frequencies of a chromatic scale (abbr: C-scale) on frequency F. In our diagrams, we have labeled the tones of the C-scale with the first twelve integers. The C-scale thus contains tones 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12. Tone 13 is written as tone 1.

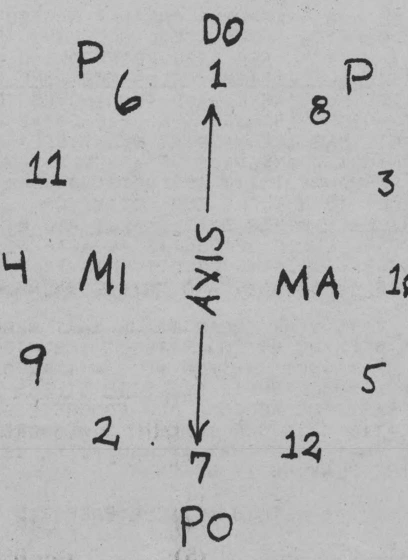


DIAGRAM B: H-SCALE

DEFINITIONS AND ABBREVIATIONS USED IN THIS ARTICLE –

C-scale: The Chromatic scale (Diagram A). Twelve tones with proportionately increasing frequencies, repeating every octave.

H-scale: The Harmonic scale (Diagram B). The eighth tone of a chromatic scale is very close to the frequency of an important overtone (the 3 overtone). The H-scale is just the C-scale redrawn to display this relationship more clearly.

DO:	the defining tone of a scale
P:	Perfect
PO:	Perfect opposite
MA:	Major
MI:	Minor

All arithmetic in C-scale music is performed "mod 12" – here's a useful mod 12 reducer

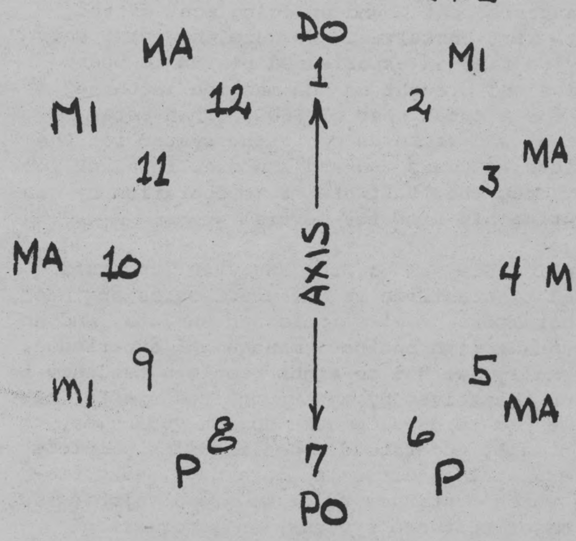
```
DEF FNM(X)=X-INT(X/12)*12
```

Here's our question for today:

Given the C-scale as the UNIVERSE, how are SUBSETS (patterns with fewer than 12 tones) with predictable properties to be selected?

Here is a simple model to help us.

DIAGRAM A: C-SCALE



CHROMATIC MAJOR,
MINOR,
AND PERFECT TONES

Chromatic tones fall into four classes:

1) *The DEFINER, and the octave of this tone (DO).* One of the 12 tones is chosen to be the "definer" of any scale. Since the C-scale can start with any tone, DO must be chosen arbitrarily. In al our diagrams in this article, DO is tone 1. The octave of DO is equivalent to DO, of course. We include the octave of DO in the scale because it makes it possible to use symmetry as a tool of analysis.

2) *MAJOR TONES (MA)* Major tones are "upward moving" with respect to DO. MA tones sound happy in a pattern (scale or melody) with DO as the definer.

On the H-scale or cycle of 5ths (see box on this page for a description of the H-scale), MA tones are all *clockwise* of DO.

If DO=1, MA tones are 3, 5, 8, 10 and 12

3) *MINOR TONES (MI).* Minor tones are "downward moving" with respect to DO. MI tones sound sad in a pattern based on DO. On the H-scale, MI tones are all *counterclockwise* of DO.

If DO=1, MI tones are 2, 4, 6, 9, and 11.

4) *PERFECT TONES (P, PO).* If DO = 1, tones 6, 7, and 8 are called *perfect* tones. This is partly traditional, partly modern.

a) Tones 6 and 8 are very close (in the C-scale) to the 2/3 and 3/2 intervals which were used by Pythagoras to define the "diatonic" (7-tone) scale (see PCC 1:3). They are honored with the title "perfect," although 6 = MI and 8 = MA by their positions on the H-scale.

b) Tone 7 is the "Perfect Opposite" (PO) of DO. When the PO appears in a pattern, the pattern is reversible, PO becoming (when it is played) a competing "DO." Tones 1 and 7 thus form an *AXIS* around which the other tones are defined. Note that MA and MI are reversed exactly when PO becomes DO. The alternation of MA and MI tones on the C-scale is interrupted by the PO.

As we shall see presently, there is a "good" reason for treating tones 6, 7, and 8 as a class.

SELECTING SEVEN-TONE SCALES

Seven-tone scales, including the familiar diatonic (major, minor) scales, are subsets of the C-scale. How is the subset specified? It turns out that there are several ways to do this. Each way leads to a somewhat different selection algorithm.

I) First, the seven tones could be selected completely at random, like dealing seven cards from a deck of 12 cards. This algorithm is left to the reader. For the moment (i.e., this article), let's look only at 7-tone patterns with somewhat even distribution (purposely omitting scales like 1 - 2 - 3 - 4 - 5 - 6 - 7 - 1).

II) *MA, MI, and P(PO).* You can use the distribution of MA, MI, and P tones to select 7-tone scales with predictable properties, based on the properties of the tones themselves. Here is one set of rules of selection:

1st location (DO)	**= 1 (always)**
2nd location	**= 2(MI) or 3(MA)**
3rd location	**= 4(MI) or 5(MA)**
4th and 5th locations	**= 6(P) or 7(PO) and 7(PO) or 8(P) [2 out of 3]**
6th location	**= 9(MI) or 10(MA)**
7th location	**=11(MI) or 12(MA)**

Diagram C shows the scale locations and which tones can occupy each location. The diagram also shows how the properties of each of the tones would affect the melodic tendencies of the scale. Major tones tend to progress upward, minor tones downward.

Why do we need a definer?

The definer (from Ta Chuan/The Great Treatise on the I Ching, or Book of Changes)

In the Book of Changes a distinction is made among three kinds of change: nonchange, cyclic change, and sequent (non-cyclic) change. Nonchange is the background, as it were, against which change is made possible. For in regard to any change there must be some fixed point to which the change can be referred; otherwise there can be no definite order and everything is dissolved in chaotic movement. This point of reference must be established, and this always requires a choice and a decision. It makes possible a system of coordinates into which everything else can be fitted. Consequently at the beginning of the world, as at the beginning of thought, there is the decision, the fixing of the point of reference. Theoretically any point of reference is possible, but experience teaches that at the dawn of consciousness one stands already enclosed within definite, prepotent systems of relationships. The problem then is to choose one's point of reference so that it coincides with point of reference for cosmic events. For only then can the world created by one's decision escape being dashed to pieces against prepotent systems of relationships with which it would otherwise come into conflict. Obviously the premise for such a decision is the belief that in the last analysis the world is a system of homogeneous relationships – that it is a cosmos, not a chaos. This belief is the foundation of Chinese philosophy, as of all philosophy. The ultimate frame of reference for all that changes is the nonchanging.

LOCATION	TONES MI	TONES MA
8	1	
7	11↓	12↑
6	9↓	10↑
5	7↓	8↑
4	6↓	7↑
3	4↓	5↑
2	2↓	3↑
1	1	

DIAGRAM C: 7-TONE SCALES

Note that selection of locations 2, 3, 6, and 7 is binary; selection of locations 4 and 5 is 2-out-of-3. No tone can be used twice. No location can be occupied by more than one tone. There are 1*2*2*3*2*2 = 48 different 7-tone scales formed by this algorithm. Choice of MA vs. MI may be independent from location to location.

In the following program (program 2), locations 2, 3, 6, and 7 are independent. Location 5 is conditional on the outcome of location 4 (see Table for Program 2). The conditional assignment of location 5 plus the need for two different formulas makes this RANDOM selection a sloppy algorithm. Can you improve it?

```
100 DIM D(7)
105 RANDØM
110 D(1)=1\D(8)=1
120 FØR I=2 TØ 7
130 R=INT(2*RND(0))
140 IF I>4 THEN 170
150 D(I)=R+2*(I-1)
160 GØ TØ 210
170 IF D(4)=6 THEN 200
180 D(5)=8
190 IF I=5 THEN 210
200 D(I)=R+2*(I-1.5)
210 NEXT I
220 PRINT
230 FØR I=1 TØ 8
240 PRINT D(I);
250 NEXT I
260 PRINT
300 END
```

(I)	R	D(I)	FORMULA	CONDITIONS
(1), (8)	not computed	1	1	always 1
(2)	1 or 0	2 or 3	R + 2*(I – 1)	none
(3)	1 or 0	4 or 5	R + 2*(I – 1)	none
(4)	1 or 0	6 or 7	R + 2*(I – 1)	none
(5)	1 or 0	7 or 8	R + 2*(I – 1.5)	D(5) = 8 if D(4) = 7
(6)	1 or 0	9 or 10	R + 2*(I – 1.5)	none
(7)	1 or 0	11 or 12	R + 2*(I – 1.5)	none

```
RUN

 1  2  4  6  7  10  12  1

READY

RUN

 1  3  5  6  7  9  11  1

READY

RUN

 1  2  5  6  7  10  11  1

READY

RUN

 1  3  4  6  8  10  11  1

READY

RUN

 1  3  4  7  8  9  12  1

READY

RUN

 1  3  5  6  7  9  11  1

READY

RUN

 1  2  5  6  7  10  11  1

READY

RUN

 1  3  4  7  8  9  12  1
```

BASIC MUSIC

Now that you've seen this program, a little question on which we shall build in later articles:

We said that there are 48 *different* scales possible by this method. This statement neglects the phenomenon of *inversion*. A scale is an inversion of another scale if

1) it uses the same *tones*, but assigns a different number the role of DO. E.g., 1 - 3 - 4 - 6 - 8 - 10 - 11 - 1 is an inversion of 11 - 1 - 3 - 4 - 6 - 8 - 10 - 11, the bracketing tones being DO in both scales.

2) the pattern of *intervals* is the same, but beginning with a different interval. We'll explain this further in a later article; try to work it out for yourself in the meantime.

How many of the 48 scales are inversions of one another? Conversely, how many really unique patterns are possible with these rules?

One further question, likewise with future significance: How many of the scales generated by this method are symmetrical? For example, 1 - 2 - 5 - 6 - 8 - 9 - 12 - 1 is symmetrical (look at Diagram C). Is this of any importance?

WRITING BID SPECS: PART II

Last issue we dealt with some general bid requirements that can be used in any computer bid situation (see box this page). This issue will deal with specifics for hardware and software. You should be aware from the start that your *software* requirements may be every bit as important as your *hardware* needs and specs should be written accordingly. Secondly, the more specific you are, the more information will be provided to you by bidders, (i.e., if you list all your requirements, it is incumbent upon the bidder to list any *exceptions* to those requirements). If you *don't* list your needs, he may not tell you all about his system and you will have to search out this information on your own.

Both hardware and software sections can and should be written with a *required* section and a *preference will be shown if you can provide this* section. This gives you wiggle room in your selection and lets your bidders know exactly what your minimum needs are and what you really want. Chances are, no one will be able to give you everything you want (at least not at a reasonable price!).

Here are some more general bid specs that have been brought to our attention since the last installment of this article. Use them in good health.

- ⋆ **The bidder must have gross sales in excess of $50 million and evidence of a profitable computer operation. [Wonder who suggested that one?] This line will certainly knock out Fly-By-Nite Manufacturing but will also knock out other small, legitimate bidders as well.**
- ⋆ **Bidder must have 10 (20, 30 ??) or more similar installations in similar institutions and must provide their names and addresses.**
- ⋆ **Bidder must have an *active* educational users group.**
- ⋆ **Bidder must provide a library of programs suitable for use in secondary schools (in BASIC).**
- ⋆ **Computer system must have a second *instructive* language (e.g., FOCAL – who claims this one?).**

HARDWARE

You can take three different approaches in writing specs for hardware. You can be SUPER-SPECIFIC. For instance, you could specify an 8K, DEC Edusystem 20 with 4 terminals (ASR 33). Unless you added "or the equivalent," this kind of spec would get you one single bidder, DEC. If you add "or the equivalent" it would be like opening Pandora's box. Everyone would bid claiming they were "equivalent" or *better* than an Edusystem 20 and you might have a real hassle proving otherwise. Unless your mind is completely closed, we don't recommend this approach.

Another approach is to spec your hardware completely around your software specs – "the hardware provided will be capable of operating the software described elsewhere in this document." This seems like an awfully gutsy thing to do and requires that your software specs be exhaustive and exacting. This approach probably makes the most sense but I'm not convinced it's practical unless you really have some sharp spec writers around your shop.

Always seeking a compromise, the obvious way to spec your system is to list those minimum hardware requirements that you think you might have plus your preferences, require that the hardware be capable of operating all the software specs and write yourself a neat set of software specs.

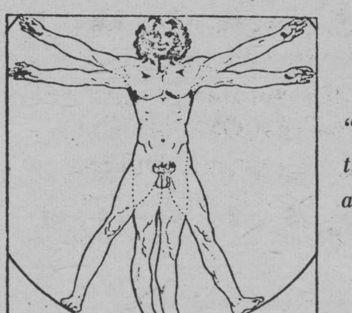
Now for the specific hardware items –

Central or Basic Computer System

Don't spec a Central Processor (CPU) per se, rather spec a total system. Require that it be new equipment (unless you want a used one) and define that it be of latest generation design (today that's third generation or is it fourth or 3½[?]). If you're gutsy don't identify your core storage requirements in terms of *x* number of words or bytes. Instead, specify your needs in *x* amount of USER SPACE or user space per terminal, while operating in time shared BASIC. (We felt 5000 words was adequate if the system could CHAIN programs.) And, if your're messing with FORTRAN and the like, then define your needs in terms of user space for each language. User space is really what you're concerned about isn't it? You will find wide variations in user space from system to system– so beware. This tactic puts the pressure on the vendor to specify space in your terms, not his. To save you dollar$ later, you should specify that the delivered system be *expandable* to *x* amount of core without the need to replace the CPU or the addition of an expansion chassis (we said 32K). It's cheaper to get the bigger chassis now and less aggravation later. If you have a preference for a 12 bit word system, say so. If you want a 16 bitter spec it that way. You may as well get what you want!

More Storage

In this day and age you have an unbelievable choice if you want more storage capability on your system. For the complete system you can choose fixed-head disks, cartridge disks, magnetic tape, DECtape, cassette tape, and a plethora of floppy disks and other assorted paraphenalia, or any combination thereof. You should examine *your* needs as *you* see them and then make your decision. We're not convinced that a fixed head disk is needed for any reason other than speed and to run up the price. If speed is not your concern, save yourself some money and don't require a fixed head disk. That doesn't mean you won't get one. Some systems only work with such a disk for reasons that have never been adequately explained. (**Warning**: If you will be doing CAI on your system you will need a fixed head disk. CAI consumes an enormous amount of disk space and it slows the system down considerably. If you plan to do anything in addition to CAI on that system, be prepared – you'll need plenty of extra disk space to handle it.)



"I'd like a computer that's about this high and this wide..."

Here are some cost savers – Most large systems are sold with magnetic tape to be used to load the "system" in case of a malfunction. This tape unit ($10,000 worth) has no other use since time sharing users *cannot* use it. You can eliminate this costly extra by using a *cartridge* disk as your time share storage unit. In case of malfunction you use this same device to reload the system.

If you're considering a DEC Edusystem, some combination of DECTAPE and cartridge disk is probably the most flexible, least cost way to go. Remember, DECTAPE is more like a random access device than it is like magnetic tape, or so they say.

Whatever you do, be sure to specify that the storage you want be available to *all* users in time shared BASIC. That seems obvious but you should know that hardware people sometimes sell you things that don't work the way you expect them to.

We're a little gun-shy about floppy disks and cassette units as of now. We haven't seen major vendors providing software to drive these units. Is that clear? Just because it's *attached* to your hardware does NOT mean it will work. There must be some linking software to make the "it" available to the user in BASIC. The software is not always available, so buyer beware . . . . However, if you have some good software people around, these two items may be a good low-cost way to adding storage to your little system.

I/O Capability

If you're specing a Teletype-only system, there is little to worry about. For each TTY you need a TTY interface or for all of them get one multiplexor into which they plug. Multiplexors usually handle 8 to 16 TTY and work out to be cheaper than buying individual interfaces. If you are going to communicate via phone lines with modems (see PCC Vol. 1, No. 2, p. 12), beware. Some TTY interfaces WILL run either direct TTY or modems – some WON'T. Specify your needs. No matter how many TTY's you want now, you'll want more soon. Be sure to require that your system I/O is expandable.

With changing technology you should be able to get multi-speed interfaces and multiplexors so you will be prepared to run those sweet new CRT's at rates of 30 CPS or faster. Your standard TTY interface will *not* run at 30 CPS. Specify variable speed interfaces or multiplexors to meet this need.

If you want card readers, printers, and all those other high speed peripherals be very cautious. These items don't just plug into the multiplexor– EACH needs a controller (like an interface). Do you want these peripherals available for time share users? You had better say so! Some BASIC systems will run these peripherals in time share, other will not. The HP 2000E, the low cost model, will not drive high-speed peripherals in time share . . . the 2000F will. (Another of life's unexplained mysteries.) Some DEC systems will drive these peripherals IF you buy more core.

Another frustrating item is the high-speed paper tape reader you must buy on larger systems for $3500 or so. It is *only* used to ENTER system software. No user can use it to enter BASIC programs (don't ask me why!).

Cost savers – Mark sense card readers – we have friends who swear by them and other who swear at them. Check mark sense readers out carefully before you decide. You may find a punch card reader will better meet your needs and save you money. Line printers are expensive as hell. A good $aving can be made if you buy the 80 column printer instead of the full blown 132 column model. Do you really need all that printing?

more memoes on "GOING TO BID"

Miscellaneous

Somewhere in your bid spec you should require that all interconnecting cables and hardware be included. Cables can run $35 to $50 each. It's nice to have them included in the price. (I realize these things sound obvious but unfortunately we know of schools that have been $-screwed by these little things.)

Most bids we've seen have specified that the system will operate without special environmental requirements such as air conditioning. You might even specify the low-high temperatures you require. That's what's neat about a mini . . . no air conditioning, no special raised floors and all those other expensive things.

Powerfail/restart capability is a hardware *and* software item. This goody costs about $500. In case of power failure or fluctuation, it guarantees you won't lose everthing that's going on and then restarts your system automatically. As a hardware item, it's pretty straightforward. Be sure to mention the requirement that there be software to make it operate when you're running BASIC. Again, it sounds obvious, but we did use a system recently that had powerfail hardware but when the plug was kicked out, we lost everything???!

Want your computer in a cabinet? Better say so, you may not get it that way.

Maintenance – the stickywicket of this business. It costs dearly but buy a yearly maintenance contract on your system and high speed peripherals. It's worth it. We don't recommend maintenance contracts on TTY's. Service them with on-call service. Be confident that you will get 12 to 24 hour service from the computer manufacturer – service by employees of the firm. I'd be very leery if maintenance is only available from a "local mechanic with whom we contract." Those of you in the boonies will encounter this problem. Beware.

We've said . . . don't buy TTY's from the computer supplier. It'll cost you a fortune. The computer supplier may require that you buy a consol TTY from them (usually an ASR 35 for $3500 or so). You're stuck buying it from them but don't accept the ASR 35 – tell them you want an ASR 33. It will do the same job at half the price.

When you specify TTY's from another source be sure to include the requirement that the TTY be modified for your system. TTY's connected to DEC, Data General and other computers, require a modification kit installed in the TTY to make it run compatibly with the computer (cost is about $100 each). Remote TTY's (via phone lines) don't require these modifications.

Leasing – Nobody buys computers these days, they lease them. Someone out there should write us an article about leasing – pros, cons, prices etc. We do know that interest rates vary from firm to firm as do all other fine print items. Let Truth-In-Lending work for you – REQUIRE the vendor to specify the interest rate used in calculating your lease.

SOFTWARE

We're only going to worry about BASIC here. If you need other language specs, you'll have to find another reference. Let me repeat the comment that this section of your specs may be more important than your hardware requirements.

BASIC was developed at Dartmouth College and there is considerable literature to explain what is called Dartmouth BASIC. Trouble is, the authors of the language have come out with several revisions and improvements to the language which have blurred the original definition. We're going to reinstate what we think is the original Dartmouth BASIC.

Dartmouth BASIC includes the following statements: LET, PRINT, READ, DATA, GOTO, IF-THEN, FOR, NEXT, GOSUB, RETURN, INPUT, REMARK, END, DEF, DIM, STOP, RESTORE, RND, SGN, SIN, COS, TAN, ATN, SQR, LOG, EXP, INT, ABS, and a full mix of MATRIX commands.

You may not need the MATRIX commands. All the rest represent the *absolute* minimum BASIC language requirements. You might place the MATRIX commands in a "desirable" software category. (MATRIX commands take up an awfully large amount of user space on core-based minis. Be sure to require the ability to delete the MATRIX commands at *your* will to gain user space when MAT isn't used. Then you only have to load the MAT commands when you need them.)

To this standard BASIC you should add the requirement of a TAB command which will help formatting output and the MULTI-BRANCH GOTO (ON x GOTO 100, 200, 300) which you will find invaluable.

If you're really getting into it, you'll want STRING VARIABLE capability on your system. String variable commands allow you to manipulate alphabetic data. Some systems will only allow strings of lengths from 6 to 18 characters, depending on the system. This is *almost* like no strings at all. HP allows strings of 72 characters (one full TTY line) on the 2000 series. DEC's BASIC PLUS language will handle strings as large as 255 characters. In addition to having strings you should specify the ability to use relational operators (<, =, >) with strings so you can do such things as compare strings and arrange them in alphabetic order. To round out your string variable capability you should require the ability to concatenate strings and separate strings using substring commands.

In the category of "preferred and awfully nice" (but not necessary) we place the ability to store strings in arrays or string array capability. This feature is now available one some of the newer, super BASIC systems that are coming out.

Next in order of preference, we see the need to CHAIN programs, that is to link two or more programs together for continuous operation. With CHAIN you must have a COMMON statement which allows carrying a variable forward from one program to another. Some systems offer CHAIN but not COMMON, caveat emptor.

The following BASIC features are not in any meaningful order but we suggest you evaluate each on its' merits and use them in your specs as you see fit, . . . as required, . . . preferred, . . . not necessary.

PRINT USING or picture formatting as a BASIC command. This gives you the ability to control output format with more precision and without some fancy programming shenanigans. Especially useful for business applications.

Multiple statements per line – DEC has a neat user space saving feature that permits you to put multiple statements on one line.

```
1Ø FOR X=1 TO 1Ø\PRINT X\NEXT X
```

If you're looking at a core based mini this is almost a necessity.

Immediate or calculator mode – allows you to execute unnumbered statements without writing a complete program and without having to scratch the existing program. The statement may be any legit BASIC statement, even including a looping one.

ENTER permits limiting the time a user has to input a value. Absolutely essential for CAI and nice to have for simulations and games.

File capability – the ability to store data in sequential and random access files. You should specify how many files can be accessed at one time (4 to 10), how much data each file should be able to handle. These figures will vary wildly from system to system.

Logical operators – AND, OR, NOT operators available to use on all data.

Peripheral drivers – if you are buying card readers etc., we repeat again, specify (REQUIRE) that the BASIC software include the ability to effectively use these items by terminal users.

There must be other BASIC features but we're out of gas! If you're seeking a large disk time sharing system than you must require a full compliment of system accounting features including *x* number of assignable user numbers, file protection for each user, and the ability to keep track of time and space for each user. You should also *require* that the system have a Public Library capability which allows all users to access the programs in this library, AND require that the vendor provide you with at least 200 programs for this library each of which be suited for educational use at your level. HP (maybe DEC too) has a nice feature called a GROUP library. Users with similar user numbers have access to this library in addition to the Public Library. This feature is appropriate in a consortium installation where each school may like its own library in addition to the Public one.

We hate to say this, but these are the recommended requirements for *one* language. If you are concerned with other languages you should take some time to specify your needs for that or those languages as well.

Is it hardware or software? Someplace you should require a reasonable response time when the system is in full use. What's reasonable??? How about 5 seconds after pressing RETURN when all 16 (or 32) users are doing normal problem-solving activities.

At the present time there is a Standards Committee meeting to establish standards for the BASIC language and all its improvements. This article has completely disregarded the existence of these standards. When they become available, we will print them so you can require the established standards on your system.

LETTERS

G. Harland
10303–98 Avenue
Edmonton Alberta T5K OC3
CANADA

Showed the paper around a lot and found that the computer people were very interested (re. the Computing Services subscription) and non-computer people were a bit bedazzled . . . if you could do something in the way of a simple analogue to help explain how a computer sort of goes about its business I think it might help.

I hesitate getting too deep into that:

(1) because the computers themselves can provide an experience worth much more than a newspaper article.
(2) for fear of the *"Radical Software Effect . . ."* I like *software* but couldn't do much with until I had a few personal exchanges with a porta-pak under by belt.

U of A has a whole mess of computers: IBM 360/6F,/40,1800, PDP8(18), PDP8E(2), HP 2116, HP 2114, D6C NOVA (3), HON DDP 316, HON DDP 516, NIC 1800, EAI 590, CDC 240, CDC L6P 30, CDC 3150, NIC 1800, XDS 920, RAY 703, UNI 9200, UNI 9300 (2), BUR L2301, but the only one I've ever spoken to is the IBM 360/67 . . . I don't even know what the rest look like or what kind of magic they perform.

No, I don't have a computer . . . do you have a 4K you can spare . . . what do you think of DEC's PDP-16 modules . . . I'm reading the book.

I'm kind of new to computers really and haven't got enough vocabulary to say much more more than a few stupid sentences but . . . I found a magazine called: SOFTWARE — Practice & Experience, Periodicals Department, John Wiley & Sons Ltd., Baffins Lane, Chichester, Sussex, ENGLAND, $23.40 USA, $22.50 CANADA/year (quarterly), that has a regular article in the back called Computer Recreations: for example,

— "Darwin" a game between computer programs as programs
— "Napolean" The Military Game
— "MOD" or
— "CALCOMP" can't read this stuff

Seymour Papert was in Edmonton a few weeks back for some seminars, etc. and mentioned:

(1) Alan Kay's involvement in some kind of computing thing (centre?) for public people in Palo Alto or wherever it is that Xerox has its mutant farm
(2) Xerox's own "computers and kids" program
(3) Ed Schlossberg (of the Brooklyn Children's Museum) and a traveling circus that involved computers, inflatables and a lot of travelling
(4) Drs. Papert & Minsky's own "computers & kids" thing @ MIT and General Turtle

Do you people know anything about these things or could you find out about them and write it up in PCC?

AN EQUAL OPPORTUNITY EMPLOYER
MINNEAPOLIS PUBLIC SCHOOLS
SPECIAL SCHOOL DISTRICT NO. 1

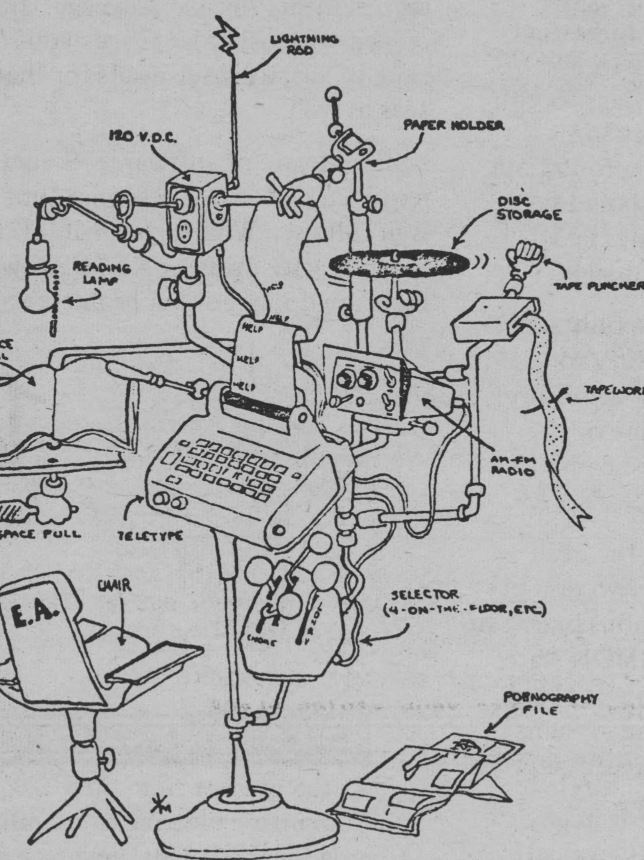
SOUTHWEST HIGH SCHOOL
3510 West 47th Street
Minneapolis, Minnesota 55410

Jim Moen, a member of the Computer Club, did this cartoon. I thought it was a clever piece of work and hope that you might publish it in a coming issue of your paper.

Jim Moen gives his consent for publication. If it is published he would like some extra copies of the issue.

Yours truly,

Edwin Andersen
Chairman, Mathematics Department



HP-2000Z
BASIC (Bad Acronyms Sicken Intelligent Computers)
TERMINAL
only $2000.00 wherever toys are sold
A GIFT FROM THE SOUTHWEST ANTI-MANAGEMENT FORCES (SAMF)

Dec 9, 72

Dear Bob

How's all the phreaks out there? I am one of Purdue University's jocks. I am a student in Engineering CS. I have a job part time at the CS center caring for the MUX. It feeds 64 TTY's and I am in the process of expanding it to 128. We run a CDC 6500 and 3-7094's on the input we have a 4K 8MOD comp with 23 ports that will be driving all our Imlac's at 9600 BAUD. (Now they are about 550 baud.) In the EE school we have a PDP-9 with a 9600 baud serial line driving the Imlac. We store program's for the Imlac on the 9's two disc's and transfer them over. We have a grad student who just finished a mini-graphical cribbage game. We run it on the '9. It's a huge fortran program.

Well I would like to get on your mailing list. I have an HP-35 and am getting parts together for a computer. (want lot's of info). Hope to hear from you. I think you guys are doing a great job.

Scientific Analysis Corporation

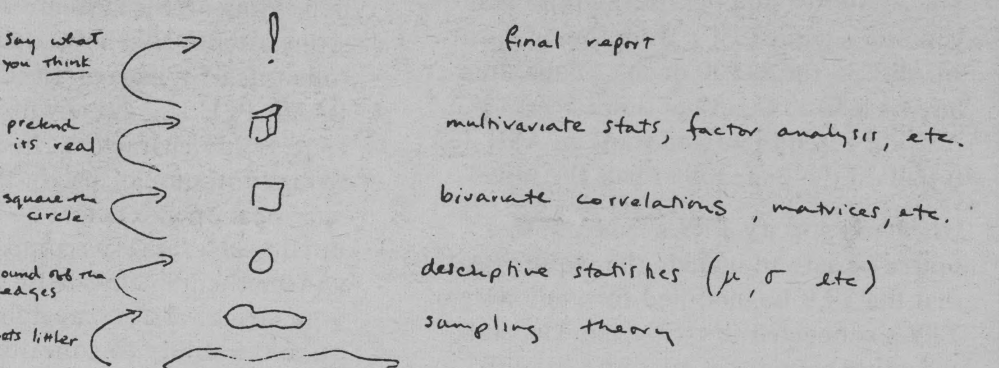
9 Feb 73

Hi!

PCC is neat.

Enclosed are some stuff I used in a seminar on computers as a research tool that I taught to our graduate sociology staff. The pictures with the trees are self-explanatory.

The "summary guide" to research design means like this, in statistical issues:


Say what you think
pretend its real
square the circle
round off the edges
gets littler
!
final report
multivariate stats, factor analysis, etc.
bivariate correlations, matrices, etc.
descriptive statistics (μ, σ etc)
sampling theory

So far all the big people I have shown these pictures to have liked them. Maybe they would work with littler people too.

Keep on trucking ..

xxx Les Morgan

A NON-PROFIT ORGANIZATION — 4339 CALIFORNIA STREET, SAN FRANCISCO, CALIFORNIA 94118 / TELEPHONE (415) 752-7180
CABLE: SPRA SAN FRANCISCO

I would like to know how to get the rest of this school year's issues. (My son and I like your magazine very much).

Sincerely,
James McCormack
James McCormack

P.S. My son, the main computer user at our house, says:

I don't have a computer, but have access to one (an HP 2000E)

I don't have a music composer, but don't really want one

The cheapest tape winder is a pair of hands

I have about five simulation/game programs I have written (and more that are either inaccuarate or not working)

I would like to see simulation/game programs that are a little more complex

I will definitely do a page in the future

PILOT '73

```
5     *MØRE T: H E R E  WE G 0 0 0 !
10    T: HEY, HEY, WHAT'S YØU SAY??
20    A:$CAT
30    T: WHØ, WHØ ARE YØUUU?
40    A:$NAME
50    T: HØW, HØW DØ YØUU FEEL L L ?
60    A:$FEEL
70    T:DØ YØU LIKE GRAPEFRUIT IN THE MØRNING??
80    A:
90    M:YES,YEAH,ØK,SURE,LØVE
100   Y:$NAME, DØES A GRAPEFRUIT FEEL $FEEL IN THE MØRNING?
110   N:I SAY, SAY $CAT TØ YØU!
120   J:*MØRE
```

```
 RUN
 H E R E  WE G 0 0 0 !
 HEY, HEY, WHAT'S YØU SAY??
?DINGBATS ARE FUNNY
 WHØ, WHØ ARE YØUUU?
?ME, HEE,HEE
 HØW, HØW DØ YØUU FEEL L L ?
?RUN ØVER BY WATERMELØNS
DØ YØU LIKE GRAPEFRUIT IN THE MØRNING??
?ILØVEIT
ME, HEE,HEE, DØES A GRAPEFRUIT FEEL RUN ØVER BY
WATERMELØNS IN THE MØRNING?

 H E R E  WE G 0 0 0 !
 HEY, HEY, WHAT'S YØU SAY??
?@
BYE
EDITØR IN
EH?10
?20
?RUN
 H E R E  WE G 0 0 0 !
 WHØ, WHØ ARE YØUUU?
?HENRY
 HØW, HØW DØ YØUU FEEL L L ?
?RUNNYNØSED
DØ YØU LIKE GRAPEFRUIT IN THE MØRNING??
?NEVER
I SAY, SAY CAT TØ YØU!
 H E R E  WE G 0 0 0 !
 WHØ, WHØ ARE YØUUU?
?@
BYE
```

P.C.C. LIVES!
PILOT IS THE PEOPLE'S LANGUAGE!
USE COMPUTERS FOR PEOPLE, NOT AGAINST THEM!

FOR THE REVOLUTIONARY

– Put some PILOT Poetry up! Find the English Department!
– Try Spanish, French, Latin (?) in PILOT!

```
        T: HOLA
```

– Invent some word games!! (Courtesy Abe Waksman, SRI)

```
        R: WORD GAME IN & OUT
        T: TROUBLE IS OUT, BUBBLE IS IN
        T: DUCK IS OUT, GOOSE IS IN
        T: FOOT IS IN, SHOE IS OUT
        T: GUESS THE RULE–GIVE ME A WORD
*MOREA:$WORD
        M: AA,BB,CC,DD,EE,FF,GG,HH,II,JJ,KK,LL,MM
        JY: *NEXT
        M:NN,OO,PP,QQ,RR,SS,TT,UU,VV,WW,XX,YY,ZZ
        N:$WORD IS OUT
        JN:*MORE
 *NEXT T:$WORD IS IN
        J:*MORE
```

```
10    T:HI! THIS IS THE GAME ØF RØØMS.
20    T:IT'S LIKE 'HIDE AND SEEK'--YØU PICK ØNE ØF THE RØØMS
30    T:ØF YØUR HØUSE, LIKE THE KITCHEN ØR
40    T:YØUR BEDRØØM
50    T:I'LL ASK YØU 'YES' ØR 'NØ' QUESTIØNS
60    T:AND TRY TØ GUESS YØUR HIDING PLACE
70    T:
80    T:PICK A RØØM AND TELL ME WHEN YØU'RE READY
90    *AGAIN T:
100   A:
110   T:ØK...
120   T:
130   T:
140   T:IS FØØD USUALLY FØUND IN YØUR RØØM
150   *ASK1 A:
160   M:YES
170   JY:*FØØD
180   M:NØ
190   JY:*NØFØØD
200   T:I'M SØØRY..
210   T:I'M SØRRY...PLEASE TYPE 'YES' ØR 'NØ'
220   J:*ASK1
230   *FØØD T:GØØD...IS IT USUALLY USED TØ EAT IN
240   *ASK2 A:
250   M:YES
260   JY:*KITCHEN
270   M:NØ
280   JY:*DININGRØØM
290   T:I'M SØRRY...PLEASE TYPE 'YES' ØR 'NØ'
300   J:*ASK2
310   *KITCHEN T:ARE YØU IN THE KITCHEN
320   A:
330   M:YES
340   JY:*GØTIT
350   JN:*GIVEUP
360   *DININGRØØM T:ARE YØU IN THE DINING RØØM
370   A:
380   M:YES
390   JY:*G3TIT
400   JN:*GIVEUP
410   *NØFØØD T:HMM...IS YØUR RØØM USUALLY USED TØ RELAX IN BY
420   T:EVERYBØDY
430   *ASK3 A:
440   M:YES
450   JY:*RELAX
460   M:NØ
470   JY:*NØRELAX
480   T:I'M SØRRY...PLEASE TYPE 'YES' ØR 'NØ'
490   J:*ASK3
500   *RELAX T:ARE YØU IN THE LIVING RØØM
510   A:
520   M:YES
530   JY:*GØTIT
540   JN:*GIVEUP
550   *NØRELAX T:DØES SØMEØNE SLEEP IN THIS RØØM
560   A:
570   M:YES
580   JY:*BEDRØØM
590   M:NØ
600   JY:*GIVEUP
610   JN:*WASH
620   *BEDRØØM T:IS IT THE BEDRØØM
630   A:
640   M:YES
650   JY:*GØTIT
660   JN:*GIVEUP
670   *WASH T:DØ PEØPLE WASH UP IN YØUR RØØM
680   A:
690   M:YES
700   JY:*BATHRØØM
710   JN:*UNDER
720   *BATHRØØM T:ARE YØU IN THE BATHRØØM
730   A:
740   M:YES
750   JY:*GØTIT
760   JN:*GIVEUP
770   *UNDER T:ARE YØU IN THE BASEMENT
780   A:
790   M:YES
800   JY:*GØTIT
810   JN:*GARAGE
820   *GARAGE T:ARE YØU IN THE GARAGE
830   A:
840   M:YES
850   JY:*GØTIT
860   JN:*GIVEUP
870   *GØTIT T:
880   T:THAT WAS FUN!
890   J:*AGAIN?
900   *GIVEUP T:I GIVE UP. WHAT RØØM ARE YØU HIDING IN
910   A:
920   T:ØH...I SEE!
930   *AGAIN? T:WANT TØ PLAY AGAIN
940   A:
950   M:Y,YES,ØK,ALRIGHT,SURE
960   JN:*ØUT
970   T:PICK ANØTHER RØØM AND TELL ME WHEN YØU'RE READY
980   J:*AGAIN
990   *ØUT E:
```

```
?RUN
HI! THIS IS THE GAME ØF RØØMS.
IT'S LIKE 'HIDE AND SEEK'--YØU PICK ØNE ØF THE RØØMS
ØF YØUR HØUSE, LIKE THE KITCHEN ØR
YØUR BEDRØØM
I'LL ASK YØU 'YES' ØR 'NØ' QUESTIØNS
AND TRY TØ GUESS YØUR HIDING PLACE

PICK A RØØM AND TELL ME WHEN YØU'RE READY

?READY
ØK...


IS FØØD USUALLY FØUND IN YØUR RØØM
?NØ
HMM...IS YØUR RØØM USUALLY USED TØ RELAX IN BY
EVERYBØDY
?YES
ARE YØU IN THE LIVING RØØM
?YES

THAT WAS FUN!
```

These PILOT 73 instructions are the "core" instructions selected by the makers of the language. Many implementations will have more instructions, some may have fewer. The idea is to keep the core set in every implementation.

A note about entering programs:

The program that makes PILOT 73 work is usually called "the Editor." This program varies in language and operation from system to system. This miniprimer is a description of the language only; we assume that you will learn about *your* Editor from the people who set up PILOT 73 on the system you will use.

TALKING COMPUTER:

To make the computer talk, you use the T: instruction (for TYPE). When you run this program

```
T:HELLØ
```

the computer says

```
HELLØ
```

(T: is just like BASIC "PRINT")

You can use the T: to make the computer print pictures!

```
T:                 +++++++++
T:               ]   -  -    [
T:              @   (0 0)     @
T:                [    A     ]
T:                 [ #=#  ]
T:                  [    ]
T:                  \\>#<//
```

Or to tell stories, etc.

```
T:THIS IS THE STØRY ØF THE DØWNFALL ØF THE NIEBELUNGS
```

(The program is left for your completion. Use extra pages if you wish.)

TALKING TO THE COMPUTER:

To get the computer to let you get in a word or two (or more), you use the A: instruction (for ANSWER). When you run this program

```
A:
```

the computer types

```
?
```

then waits for your response. After you have typed in something and pressed the RETURN key, the computer will continue with the rest of the program (if any).

Here's a short program

```
T:THIS IS THE QUESTIØN
T:WHAT IS YØUR BIRTHSIGN
A:
T:THAT'S NICE
```

When you run this program, here's what happens:

```
THIS IS THE QUESTIØN
WHAT IS YØUR BIRTHSIGN
?
```

Then the computer waits for your answer . . . . If you then type

```
CAPRICØRN
```

(followed, of course, by the RETURN key)

the computer continues with

```
THAT'S NICE
```

Now, you do it: write a PILOT program using T: and A: which "behaves" like the interviewer on a TV "talk show" (like, Cavett, Carson, etc.). That is, it asks question after question, without paying attention to the answers at all.

Or, if that's not your cup of T:, try simulating Mother Goose. Here's a typical MG, telling a story to her children

```
WHEN SHE HEARD THIS, THE WICKED WITCH JUMPED ØN HER
BRØØMSTICK, AND HEADED AS FAST AS SHE CØULD FLY BACK
TØ HER HØUSE IN THE WØØDS.
AND WHAT DØ YØU THINK SHE DID WHEN SHE GØT THERE
?ATE UP THE KIDS
?NØ, SHE JUST CHASED THEM        MG's children reply
?OOOO, THE WICKED WITCH!
WELL, WHEN SHE GØT THERE, THE FIRST THING SHE DID WAS...
```

(Well, what was the first thing she did?)

THE COMPUTER TALKS SOME MORE:

```
WHAT IS YØUR NAME
?FEARLESS FRED
HELLØ, THERE, FEARLESS FRED
```

How did the computer do that? By using an **answer label**. *Here's the program:*

```
T:WHAT IS YØUR NAME
A:$NAME                this is an answer label
T:HELLØ, THERE, $NAME
```

Let's look at some more computer talk:

```
T:WHAT'S YØUR NAME
A:$NAME
T:HØW ØLD ARE YØU (IN YEARS)
A:$AGE
T:WHERE DØ YØU LIVE
A:$LIVE
T:SØ, YØUR NAME IS $NAME, AND YØU LIVE IN $LIVE
T:AND YØU ARE $AGE YEARS ØLD.
```

See how answer labels work? The $ means "the next characters are an **answer label**.*" When the computer sees*

```
A:$NAME
```

it labels the response with the label "$NAME." Then when it sees:

```
T:YØUR NAME IS $NAME
```

the computer types the first part

```
YØUR NAME IS
```

then it looks for the A: line which contains the label $NAME. If it can find such a line, it will type the response which was given there. If it can't find the line, or if that line hasn't been reached yet (in the program), the computer types the label.

```
WHAT'S YØUR NAME
?CLEØPATRA
HØW ØLD ARE YØU (IN YEARS)
?2500
WHERE DØ YØU LIVE
?EGYPT
SØ, YØUR NAME IS CLEØPATRA, AND YØU LIVE IN EGYPT
AND YØU ARE 2500 YEARS ØLD.
```

What next? JUMPING!

```
*HERE T:I AM HERE
T:I AM THERE
J:*HERE
```

(This is a line label)

When you RUN this, the computer says:

```
I AM HERE
I AM THERE
I AM HERE
I AM THERE
I AM HERE
```

etc., until you stop the computer (BREAK key on some systems). You use a **line label** *(like *HERE, *LABEL, *START, etc.) to tell the J: instruction (for JUMP) where to jump.*

```
*HERE T:I AM HERE
J:*THERE
*WHERE T:WHERE AM I?
J:*HERE
*THERE T:I AM THERE
J:*WHERE
```

will produce the same result as

```
*BEGIN T:I AM HERE
T:I AM THERE
T:WHERE AM I?
J:*BEGIN
```

What is the result?

Here's a "fancy" way to jump (or, here come subroutines):

```
*BEGIN T:HERE'S THE START
U:*MIDDLE
T:THIS IS THE END
J:*END
*MIDDLE T:THIS IS THE MIDDLE
E:
*END T:GØØDBYE
E:
```

U:

```
?RUN

HERE'S THE START
THIS IS THE MIDDLE
THIS IS THE END
GØØDBYE
```

So,

U: (for USE) acts just like J: (jumps to a label)

E: (for END) does two things

*(1) Ends the whole program (like BASIC "END"). The last E: in the program above does this, right after the label *END.*

(2) If a U: line has been executed, E: **returns** *the computer to the line following the U:. The first E: above, right after the label *MIDDLE, does this. When the computer reaches this instruction, it automatically jumps back to the line right after the U: instruction.*

M:

THE WORLD OF M:

The M: instruction (for MATCH) is used to make the computer a little "smarter." LOOK HERE

```
T:DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
A:
M:EVØLUTIØN,SPECIES,BEAGLE
TY:FØUND A MATCH!
TN:NØ MATCH

?RUN

DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
?
```

Now, if the responder types in one of the three M: items, a "match" will be found.

If a match is found, the value of M: is "YES" (Y).

If no match is found, the value of M: is "NO" (N).

M: executes a **character by character** *string comparison of the response given to A:, with each of the M: items, one item at a time. (In other words, a "moving window" comparison.)*

M: takes everything literally. Suppose we run the above program.

```
DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
?NØ◄———— an honest answer. . .
NØ MATCH
```

N

M: is NO.

Again:

```
DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
?INVENTED SPESES
NØ MATCH
```

(M: is NO 'cause the responder can't spell.)

Once more:

```
DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
?THEØRY ØF EVØLUTIØN
FØUND A MATCH!
```

Y

(M: is YES, since the A: matches one of the M: items.)

To get the computer to "ignore" irrelevant spelling errors and/or extra text, the M: items may consist of a few consecutive characters of the desired response, for example:

```
M:EVØL,SPE,BEAG
```

You may get in a little trouble with this approach. Consider:

```
DØ YØU KNØW WHAT DARWIN IS FAMØUS FØR
?THE SEXUAL REVØLUTIØN
```

If the M: line is

```
M:EVØL,SPE,BEAG
```

then M: **will** *find a match, because REVOLUTION contains one of the M: items.*

This should help you to figure out how M: works. That's only the first part. Next, let's do something with the value of M:. Look at this program:

```
T:WHAT AMERICAN WAS THE 'FATHER ØF HIS CØUNTRY'
A:
M:GEØ,WASH
TY:GØØD FØR YØU, SMARTY.
JY:*END
T:SØRRY, THAT'S NØT IT
*END E:
```

If M: is Y after execution, the instructions

```
TY:GØØD FØR YØU, SMARTY.
JY:*END
```

will be performed. If no match is found, M: will be N, and all Y: instructions will be skipped. Here's the RUN:

```
WHAT AMERICAN WAS THE 'FATHER ØF HIS CØUNTRY'
?SAM SPADE
SØRRY, THAT'S NØT IT
```

Let's try again

```
WHAT AMERICAN WAS THE 'FATHER ØF HIS CØUNTRY'
?WASHINGTØN
GØØD FØR YØU, SMARTY.
```

*Because M: is YES, the JY: instruction jumps to the line labeled *END.*

Any PILOT 73 instruction can be make "conditional" on the **last** *M: line executed by adding Y or N to the instruction.*

Examples:

```
AY  JN  MN  UY  EN  MY
```

A few other details:

(1) M: looks at the last A: executed.

(2) M: lines can contain as many items as you can fit in a line.

(3) TY: can be written as just Y:

(4) TN: can be written as just N:

(5) Each time a new M: line is executed, the value (that is, Y or N) of any previous M: line is erased.

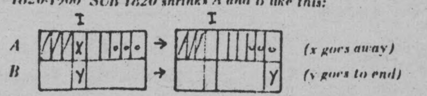
Example:

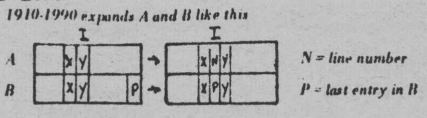
```
T:WHERE IS THE LARGEST STØCK MARKET LØCATED,
T:NEW YØRK, MØSCØW, ØR LØNDØN
A:
M:NEW,YØRK,NY
Y:RIGHT ØN
JY:*END
M:MØS
Y:SØRRY, THE RUSSIANS AREN'T CAPITALISTS (YET)
JY:*END
MN:LØN
Y:SØRRY, THE SUN HAS SET ØN THE EMPIRE
*END E:

?RUN

WHERE IS THE LARGEST STØCK MARKET LØCATED,
NEW YØRK, MØSCØW, ØR LØNDØN
?MØSCØW
SØRRY, THE RUSSIANS AREN'T CAPITALISTS (YET)

?RUN

WHERE IS THE LARGEST STØCK MARKET LØCATED,
NEW YØRK, MØSCØW, ØR LØNDØN
?LØNDØN
SØRRY, THE SUN HAS SET ØN THE EMPIRE
```

PILOT 1

By Gregory Yob

```
PILØT1

5   CØM K$[1]
10  REM-PILØT 73 SYSTEM BX GREGØRY YØB
5   REM-2296 BRYANT PALØ ALTØ CAL. 326-4039
20  I=1
10  REM- CØPYRIGHT,1972
10  DIM A$[41],B$[41],C$[41],D$[41],E$[41],F$[41]
0   DIM S$[72],Ø$[72]
0   DIM T$[72]
0   REM-SET PO TØ LENGTH ØF S$ AND Ø$ IN ABØVE   PO is maximum string length
0   PO=72                                         allowed on your system
0   DIM A[250],B[250],L[50],M[50]
00  DEF FNA(X)=X-INT((X-1)/6)*6         FNA chooses which string in a record
10  DEF FNB(X)=INT((X-1)/6)+1           FNB chooses which record to access
20  IF K$#"" THEN 190                   A string in common is initialized to NVL.  This detects
30  PRINT  INSTRUCTIØNS";               whether a chained program is RUN or not previously
40  INPUT S$
50  IF S$[1,1]#"Y" THEN 190
60  CHAIN "PILØT3"                      "PILOT 3" has instructions & chains back to "PILOT 1"
90  FILES TEST,E                        Always different in each dialect of BASIC
00  IF  END #1 THEN 260                 Checks whether to initialize (SCRATCH) your file (TEST)
10  READ #1,1
20  MAT  READ #1;A
30  READ #1,5                           Absorb A and B arrays
40  MAT  READ #1;B
50  GØTØ 310
60  GØSUB 5690                          SUB 5690 initializes the file TEST
00  PRINT "SCRATCH PERFØRMED ØN FILE"                 Just a reminder
10  REM-EDITØR PRØGRAM
20  PRINT "EDITØR IN"                   First thing you see in RUN or returning from PILOT 2
30  PRINT "EH";                         Indicates illegal command
40  INPUT S$                            S$ holds your input for parsing, etc.
50  I6=0
60  GØSUB 5410                          SUB 5410 tries to get a line number  N= -1 = no numbers found
70  IF N<1 THEN 2000                    Branch to [RUN,REN,SCR,LIS]
80  REM-NUMERIC CØMMANDS
90  IF S$[1,2]="??" THEN 700
00  IF S$[1,1]="?" THEN 900             Line 700 lists 10 lines.   Line 900 lists one line..
10  IF S$#"" THEN 1000                  If only a number, delete
20  REM-DELETE A LINE                   DELETE LINES
30  FØR I=1 TØ 240
40  IF A[I]=N THEN 470
50  NEXT I                              Locate line number if double length
60  GØTØ 340
70  IF A[I+1]=N THEN 580
80  REM-SINGLE LINE
90  P=B[I]
00  FØR J=I TØ 239
10  A[J]=A[J+1]
20  B[J]=B[J+1]                         Single line deletion
30  NEXT J
40  A[240]=0
50  B[240]=P
60  GØSUB 5200                          SUB 5200 saves A and B on disk
70  GØTØ 340
80  REM-DØUBLE LINE
90  P=B[I]
00  P1=B[I+1]
10  FØR J=I TØ 238
20  A[J]=A[J+2]                         Double line deletion
30  B[J]=B[J+2]                         Deletion involves, bumping A array, bumping B arrays
40  NEXT J                              and saving address at end of B array (or available space
50  A[240]=A[239]=0                       shrinks.
60  B[239]=P
70  B[240]=P1
80  GØSUB 5200
90  GØTØ 340
700  REM-LIST TEN LINES                 700-880 lists 10 lines
710  FØR I=1 TØ 240
720  IF A[I] >= N THEN 750              Locate line number
730  NEXT I
740  GØTØ 340
750  PRINT ""                           "" is for N^C line feed in HP BASIC
760  J1=0
770  IF J1>10 THEN 870
780  GØSUB 5270                         SUB 5270 looks up the string
790  IF A[I+1]#A[I] THEN 820            Check if double length
800  I=I+1
810  IF S$="" THEN 870
820  PRINT A[I];S$
830  I=I+1
840  IF A[I]<1 THEN 870
850  J1=J1+1
860  GØTØ 770
870  PRINT ""                           "" is for NC line feed in HP BASIC
880  GØTØ 340
890  REM-ØNE LINE                       890-990  List one line
900  FØR J=1 TØ 240
910  IF A[J]=N THEN 940
920  NEXT J
930  GØTØ 340
940  PRINT ""                           "" is for line feeds
950  I=J
960  GØSUB 5270
970  PRINT A[I];S$
980  PRINT ""
990  GØTØ 340
1000  REM- INSERT ØF ØVERWRITE          Save or overwrite a line (hairiest part of editor)
1010  FØR I=1 TØ 240
1020  IF A[I]=N THEN 1080               Check if overwrite
1030  IF A[I]>N THEN 1400               Check if last line
1040  IF A[I]=0 THEN 1570               Insertion
1050  NEXT I
1060  PRINT "NØ MØRE SPACE"             Error flag
1070  GØTØ 340
1080  REM-ØVERWRITE                     1080-1390 Overwrite - Possible cases:
                                                                SINGLE    DOUBLE
1090  IF I+1>240 THEN 1110
1100  IF A[I]=A[I+1] THEN 1250          Single length input     1110      1270
1110  REM- ØNE LINE IN FILE             Double length input     1150      1320
1120  IF LEN(S$)>41 THEN 1160
1130  GØSUB 5040
1140  GØTØ 340
1150  REM- TWØ LINE INSERT, ØNE LINE IN FILE
1160  T$=S$
1170  S$=S$[1,41]
1180  GØSUB 5040
1190  I=I+1
1200  S$=T$[42]
1210  GØSUB 1910                        SUB 1910 bumps A and B (makes a hole)
1220  GØSUB 5040                        SUB 5040 stores string on disk
1230  GØSUB 5200
1240  GØTØ 340
1250  REM- TWØ LINES IN FILE
1260  IF LEN(S$)>41 THEN 1320
1270  REM- ØNE LINE TØ PUT INTØ TWØ LINE FILE
1280  GØSUB 1820                        SUB 1820 anti-bumps A and B (fills a hole)
1290  GØSUB 5040
1300  GØSUB 5200                        Saves A and B arrays
1310  GØTØ 340
1320  REM- TWØ AND TWØ
1330  T$=S$
1340  S$=S$[1,41]
1350  GØSUB 5040
1360  I=I+1
1370  S$=T$[42]
1380  GØSUB 5040
1390  GØTØ 340
1400  REM- INSERTIØN                    1400-1560  Insert new line between others
1410  GØSUB 1910
1420  IF LEN(S$)>41 THEN 1470           Check if double length
1430  REM- SINGLE INSERT
1440  GØSUB 5040
1450  GØSUB 5200
1460  GØTØ 340
1470  REM- DØUBLE INSERTIØN
1480  T$=S$
1490  S$=S$[1,41]
1500  GØSUB 5040
1510  I=I+1
1520  S$=T$[42]
1530  GØSUB 1910
1540  GØSUB 5040
1550  GØSUB 5200
1560  GØTØ 340
1570  REM- LAST LINE CASE               1570-1740  If new line equal to last line then this
1580  IF LEN(S$)>41 THEN 1650           block is used
1590  REM- ØNE LINE, PLEASE             Single szie
1600  A[I]=N
1610  GØSUB 5040
1620  GØSUB 5200
1630  GØTØ 340
1640  REM-DØUBLE IF YØU WISH            Double size
1650  A[I]=N
1660  A[I+1]=N
1670  T$=S$
1680  S$=S$[1,41]
1690  GØSUB 5040
1700  S$=T$[42]
1710  I=I+1
1720  GØSUB 5040                        Store string
1730  GØSUB 5200                        Save A and B arrays
1740  GØTØ 340
1820  REM- GIVEN I< DELETE THE LINE
1830  P=B[I]                            1820-1900  SUB 1820 shrinks A and B like this:
1840  FØR J=1 TØ 239
1850  A[J]=A[J+1]
1860  B[J]=B[J+1]
1870  NEXT J
1880  A[240]=0
1890  B[240]=P
1900  RETURN
1910  REM- GIVEN I , BUMP ØUT ØNE LINE
1920  P=B[240]                          1910-1990 expands A and B like this
1930  FØR J=240 TØ I+1 STEP -1
1940  A[J]=A[J-1]
1950  B[J]=B[J-1]
1960  NEXT J
1970  A[I]=N
1980  B[I]=P                            (As you may note, insertion & deletion do strange things)
1990  RETURN
```





```
2000  REM-NØN NUMERIC CØMMANDS          Recall Line 370 - this is the other main branch of the editor
2010  S$=S$[1,3]
2020  IF S$="SCR" THEN 2070
2030  IF S$="LIS" THEN 2100             Look at 1st 3 characters of S$ and branch
2040  IF S$="REN" THEN 2220
2050  IF S$="RUN" THEN 2360
2060  GØTØ 330                          Return to EH?, Line 330 if not recognizable
2070  REM- SCRATCH
2080  GØSUB 5690
2090  GØTØ 340
2100  REM-LIST                          List
2110  PRINT ""
2120  FØR I=1 TØ 240
2130  IF A[I]<1 THEN 2190
2140  GØSUB 5270
2150  PRINT A[I];S$""                   The "" contains X^c (turns punch off) which allows the
2160  IF A[I]#A[I+1] THEN 2180          tape to be used as input at a later time
2170  I=I+1
2180  NEXT I
2190  PRINT ""
2200  GØTØ 340
2210  REM- RENUMBER                     Renumber by 10's  (I5 holds increment value)
2220  I5=10
2230  FØR I=1 TØ 240
2240  IF A[I]<1 THEN 2340               If zero, quit
2250  IF A[I]=A[I+1] THEN 2290          Check if double length
2260  A[I]=I5
2270  I5=I5+10
2280  GØTØ 2330
2290  A[I]=I5
2300  A[I+1]=I5
2310  I=I+1
2320  GØTØ 2270
2330  NEXT I
2340  GØSUB 5200                        Save A and B arrays
2350  GØTØ 340
2360  CHAIN "PILØT2"                    RUN (PILOT 2 RUNs the PILOT program)
5040  REM-GIVEN I, STASH S$             Messy as you can see.  Read a record, rewrite holding new strings S$
5050  V=FNB(B[I])+8                     Because strings start at record 9 in TEST
5060  READ #1,V;A$,B$,C$,D$,E$,F$
5070  GØTØ FNA(B[I]) ØF 5080,5100,5120,5140,5160,5180
5080  PRINT #1,V;S$,B$,C$,D$,E$,F$
5090  RETURN
5100  PRINT #1,V;A$,S$,C$,D$,E$,F$
5110  RETURN
5120  PRINT #1,V;A$,B$,S$,D$,E$,F$
5130  RETURN
5140  PRINT #1,V;A$,B$,C$,S$,E$,F$
5150  RETURN
5160  PRINT #1,V;A$,B$,C$,D$,S$,F$
5170  RETURN
5180  PRINT #1,V;A$,B$,C$,D$,E$,S$
5190  RETURN
5200  REM-PRINT A AND B ARRAYS          No comment.  Save A and B arrays
5210  READ #1,1
5220  MAT  PRINT #1;A
5230  READ #1,5
5240  MAT  PRINT #1;B
5250  RETURN
5270  REM- EXTRACT S$ WHEN GIVEN I   I ≠ string address.  Get S$ given I
5280  READ #1,FNB(B[I])+8               One mile!!!  (Note the 8)
5290  FØR J=1 TØ FNA(B[I])
5300  READ #1;S$                        Serial READ works fine.  If you are a time fanatic, note 5280
5310  NEXT J                            which gets you nearby
5320  IF I+1>240 THEN 5400              Check if file is full
5330  IF A[I+1]#A[I] THEN 5400          Check for double length
5340  I6=B[I+1]
5350  READ #1,FNB(I6)+8
5360  FØR J=1 TØ FNA(I6)
5370  READ #1;A$
5380  NEXT J
5390  S$[LEN(S$)+1]=A$                  Concatenations in HP BASIC
5400  RETURN
5410  REM-GIVEN S$ GET N                Convert string "9" to numeric 9.0 etc.  Gets line number
5419  DIM Z$[10]
5420  Z$="0123456789"
5430  N=-1
5440  GØSUB 5550                        5440-5530  SUB 5550 removes leading blanks
5450  IF S$="" THEN 5540
5460  GØSUB 5610                        compare digit with numeral
5470  IF N1<0 THEN 5530
5480  IF N>-1 THEN 5500
5490  N=0
5500  N=10*N+N1                         Build N
5510  S$=S$[2]                          Clip off first character in S$
5520  GØTØ 5450
5530  GØSUB 5550
5540  RETURN
5550  REM-DEBLANK S$                    Look 'n see!
5560  IF S$[1,1]#" " THEN 5600
5570  S$=S$[2]
5580  IF S$="" THEN 5600
5590  GØTØ 5560
5600  RETURN
5610  REM-DIGIT SCAN                    Set N1 = value of digit 0 - 9
5620  N1=-1
5630  FØR J=1 TØ 10
5640  IF S$[1,1]#Z$[J,J] THEN 5670
5650  N1=J-1
5660  RETURN
5670  NEXT J
5680  RETURN
5690  REM-SCRATCH RØUTINE               Set A = 0, B = 1, 2, 3, . . . , 240
5700  MAT A=ZER
5710  MAT B=ZER
5720  FØR J=1 TØ 240
5730  B[J]=J
5740  NEXT J
5750  A$=""
5760  FØR J=9 TØ 48                     Print NUL strings in $STRING$ portion
5770  PRINT #1,J;A$,A$,A$,A$,A$,A$
5780  NEXT J
5790  GØSUB 5200                        Saves statements
5800  RETURN
6100  END
```

NOTE: PILOT 1 and PILOT 2 have compatible line numbers and can be joined into one program if you care to dare. (Lines 2400-5000 belong to PILOT 2)

PILOT 2

By Gregory Yob

```
PILØT2

5  CØM K$[1]                            Lines 5 to 200  preliminaries
20  I=1
30  K$="Y"
40  DIM A$[41]
50  DIM S$[72],Ø$[72]
80  PO=72
90  DIM A[241],B[241],L[50],M[50]
100  DEF FNA(X)=X-INT((X-1)/6)*6
110  DEF FNB(X)=INT((X-1)/6)+1
190  FILES TEST,E
200  GØTØ 2400
320  CHAIN "PILØT1"                     Exit to editor
2400  RESTØRE                           Initializing
2410  GØSUB 4680                        SUB 4680 initializes lots of stuff
2420  I8=0                              I8, I9 and Q are counters
2430  I9=0
2440  READ #1,1
2450  MAT  READ #1;A
2460  READ #1,5
2470  MAT  READ #1;B
2480  MAT M=ZER                         Here we start the first pass (assemble symbol tables)
2490  MAT L=ZER
2510  F=0
2520  Q=1
2530  FØR I=1 TØ 240                    Scan loop.  I = all string addresses in $STRINGS$
2535  GØSUB 5280                        SUB 5280  Given I, return S$ from $STRINGS$
2536  Q0=LEN(S$)
2540  IF S$="" THEN 2750                If NUL string, first pass is complete.  Start execution
2545  IF S$[1,1]#"*" THEN 2585 Check for *label.  Branch around if not
2555  GØSUB 3690                        SUB 3690 extracts * label as D$
2560  GØSUB 4920                        SUB 4920 hash codes D$ as Q1
2565  L[Q]=Q1                           Store hashed lable in L
2570  M[Q]=I                            Store address in M
2575  Q=Q+1                             Increment counter Q
2580  IF Q>50 THEN 2750                 Test if table full.  If so, begin execution     S$ is now = text to right of colon
2585  GØSUB 3580                        SUB 3580  Look for colon in statement.  C$ = command (TYNABM)
2590  IF C$[1,1]#"A" THEN 2640 Skip if not A:
2595  IF S$[1,1]#"$" THEN 2640 Skip if no $ variable to save
2600  GØSUB 3690                        Get $ varible as D$, hash code D$
2605  GØSUB 4920
2610  I9=I9+1                           Increment counter I9
2615  IF I9>20 THEN 2640                If table is full, ignore
2620  S[I9]=Q1                          Save in table "S"
2625  GØTØ 2565
2630  IF Q0<42 THEN 2640
2635  I=I+1                             Increment counter I (address) according to single or
2640  IF Q0<42 THEN 2650                double line
2645  I=I+1
2650  NEXT I                            End of block
2750  I=0                               Begin execution  I = program counter
2760  I=I+1                             Increment program counter!
2770  IF I8<1 THEN 2790                 It checks for double length line
2780  I=I+1
2790  I8=0
2800  IF I>240 THEN 3490                Escape if end of file
2810  GØSUB 5280                        Get new S$, test if NUL, test if double length, set I8 flag
2820  IF S$="" THEN 320
2830  IF LEN(S$)<42 THEN 2860
2840  I8=1
2860  IF S$[1,1]#"*" THEN 2920
2870  GØSUB 3690
2880  IF LEN(S$)>0 THEN 2920
2890  GØTØ 320
2920  GØSUB 3580                        Get G$ = condition,  C$ = command
2935  IF G$="Y" THEN 3000               Check if G$ exists
2940  IF G$="N" THEN 3020
2945  C$=C$[1,1]
2950  IF C$="J" THEN 3360  ┐
2955  IF C$="A" THEN 3220  │
2960  IF C$="M" THEN 3190  ├──────── Branch to command
2961  IF C$="R" THEN 2760  │
2965  IF C$="E" THEN 2890  ┘
2970  GØTØ 3140                         T: default
3000  IF F>0 THEN 2945                  Check if "yes"
3010  GØTØ 2760
3020  IF F<1 THEN 2945                  Check if "no"
3030  GØTØ 2760
3140  GØSUB 3820                        SUB 3820 scans for $ variable and prints text
3150  T=T+1                             T is a loop flag.  If > 100, will exit program
3160  IF T>100 THEN 3470
3170  GØTØ 2760                         Back to main loop
3190  GØSUB 4520                        Match routine SUB 4520
3200  GØTØ 2760
3220  INPUT I$                          A:
3222  I$[LEN(I$)+1]=""     ┐
3225  I$[4,LEN(I$)+3]=I$   ├──────── Add leading and trailing blanks to I$
3227  I$[1,3]=""           ┘
3230  T=0                               Turn off loop counters (reset to zero)
3240  IF I$[4,4]="@" THEN 3490 Check if "@" abort (chain to PILOT 1)
3250  T1=T=0
3255  IF S$[1,1]#"$" THEN 3340
3260  GØSUB 3690                        Save reply in "E"   3280 – find address
3265  GØSUB 4920
3280  FØR J=1 TØ I9
3290  IF Q1=S[J] THEN 3320
3300  NEXT J
3310  GØTØ 2760
3320  L2=J
3330  GØSUB 4370                        SUB 4370 stores reply in "E"
3340  GØTØ 2760
3360  T1=T1+1                           Jumps – increment jump counter, test if greater than 100
3370  IF T1>100 THEN 3450
3375  D$="*"                            A fudge fixit!
3380  GØSUB 3690                        Extract * label    SUB 4120 looks for address
3390  GØSUB 4120
3400  IF M2=0 THEN 3430                 If M2 (new address) = 0, label not found in table
3410  I=M2                              Set program counter and jump!
3420  GØTØ 2800
3430  PRINT "CANNØT LØCATE '"D$"'"  ┐
3440  GØTØ 2760                     │  Error messages
3450  PRINT "JUMP LØØP AT "D$       │
3460  GØTØ 3490                     ┘
3470  PRINT "TEXT LØØP"
3490  PRINT "BYE"                       Exit program to editor PILOT 1
3500  GØTØ 320
3520  IF S$[1,1]#" " THEN 3570 SUB 3520 remove leading blanks
3530  IF LEN(S$)<1 THEN 3570
3540  S$=S$[2]
3550  IF LEN(S$)<2 THEN 3570
3560  GØTØ 3520
3570  RETURN
3580  GØSUB 3520                        SUB 3580  look for ":"   C$ = command   G$ = condition
3610  FØR J1=2 TØ LEN(S$)
3620  IF S$[J1,J1]=":" THEN 3650
3630  NEXT J1
3640  RETURN
3650  C$=S$[1,1]
3655  G$=S$[J1-1,J1-1]
3660  S$=S$[J1+1]
3670  RETURN
3690  GØSUB 3520                        SUB 3690  Deblank and get * label or $ variable D$ from S$
3695  D$=" "
3710  FØR J1=2 TØ LEN(S$)
3711  IF S$[J1,J1]="$" THEN 3740  ┐
3712  IF S$[J1,J1]="*" THEN 3740  │
3713  IF S$[J1,J1]<"0" THEN 3790  ├──── Legal character set.  If illegal character, end-of-label
3714  IF S$[J1,J1]<":" THEN 3740  │
3720  IF S$[J1,J1]<"A" THEN 3790  │
3730  IF S$[J1,J1]>"Z" THEN 3790  ┘
3740  NEXT J1
3750  J2=LEN(S$)
3755  DIM D$[10]
3760  REM
3765  D$[1,10]=S$[1,J2]
3770  S$=S$[J2+1]
3780  RETURN
3790  J2=J1-1
3800  GØTØ 3760
3820  P=1                               SUB 3820  Scan text for $ variable.  Print text and $ variable
3830  P1=LEN(S$)
3840  IF S$[P,P]="$" THEN 3900 Test if character is "$"
3845  PRINT S$[1,1];                    Print a character
3846  IF LEN(S$)<2 THEN 3880
3850  S$=S$[2]
3870  GØTØ 3830
3880  PRINT                             Carriage and line feed when finished
3890  RETURN
3900  IF S$[P+1,P+1]=" " THEN 3850      Rejects "$_" as a $ variable.  O$ is a temporary string
3910  Ø$=S$
3920  S$=S$[P]
3930  GØSUB 3690                        390-4010  Locate X$ as $ variable from "E"
3940  GØSUB 4920
3950  L2=0
3960  FØR J=1 TØ I9
3970  IF Q1#S[J] THEN 3990
3980  L2=J
3990  NEXT J
4000  IF L2#0 THEN 4010
4003  PRINT "$";
4006  GØTØ 4090
4010  GØSUB 4240                        SUB 4240 get X$ from "E"
4020  Ø$[P]=X$[3,LEN(X$)-3]             Given hashcoded * label as Q1, try to find it
4030  P1=LEN(Ø$)
4040  IF P1+LEN(S$)<PO THEN 4080
4050  Ø$[LEN(Ø$)+1]=S$[LEN(D$)+1]
4060  S$=Ø$
4070  GØTØ 3880                         If not found, error and set to program counter +1
4080  Ø$[LEN(Ø$)+1]=S$
4090  S$=Ø$
4100  GØTØ 3850
4120  REM
4130  GØSUB 4920
4140  FØR J=1 TØ 50
4150  IF L[J]=Q1 THEN 4200
4160  NEXT J
4170  PRINT D$" IS MISSING"
4180  L2=M2=I+1
4190  RETURN
4200  I=M[J]
4210  L2=M2=I
4220  RETURN
4240  L3=FNC(L2)                        Given L2, get X$ from "E"
4250  READ #2,L3;W$,X$,Y$,Z$
4260  L4=FND(L2)
4270  IF L4<2 THEN 4340
4280  IF L4<3 THEN 4310
4290  IF L4<4 THEN 4320
4300  X$=Z$
4310  RETURN
4320  X$=Y$
4330  RETURN
4340  X$=W$
4350  RETURN
4370  L3=FNC(L2)                        Given L2, put X$ into "E"
4380  READ #2,L3;W$,X$,Y$,Z$
4390  L4=FND(L2)
4400  IF L4<2 THEN 4490
4410  IF L4<3 THEN 4470
4420  IF L4<4 THEN 4450
4430  PRINT #2,L3;W$,X$,Y$,I$
4440  RETURN
4450  PRINT #2,L3;W$,X$,I$,Z$
4460  RETURN
4470  PRINT #2,L3;W$,I$,Y$,Z$
4480  RETURN
4490  PRINT #2,L3;I$,X$,Y$,Z$
4500  RETURN
4520  S$[LEN(S$)+1]=","                 MATCH FUNCTION
4521  FØR J2=2 TØ LEN(I$)
4522  IF I$[J2,J2]#I$[J2-1,J2-1] THEN 4526
$523  IF I$[J2,J2]#" " THEN 4526        Remove multiple blanks in answer
4524  I$[J2-1]=I$[J2]
4525  IF J2=LEN(I$) THEN 4530
4526  NEXT J2
4530  F=0                               Flag = "N"
4540  IF LEN(S$)<2 THEN 4660            No more cue words – branch
4550  FØR J1=1 TØ LEN(S$)
4560  IF S$[J1,J1]="," THEN 4580
4570  NEXT J1
4580  X$=S$[1,J1-1]                     Get X$ as cue word.  Remove leading and trailing blanks
4581  IF X$[1,1]#" " THEN 4585
4582  X$=X$[2]
4583  IF X$="" THEN 4530
4584  GØTØ 4581
4585  IF X$[LEN(X$),LEN(X$)]#" " THEN 4590
4586  X$=X$[1,LEN(X$)-1]
4587  IF X$="" THEN 4530                Blank cue word exit
4588  GØTØ 4585
4590  S$=S$[J1+1]                       Truncate S$ for next cue word
4600  IF LEN(X$)>LEN(I$) THEN 4540
4610  FØR J1=1 TØ LEN(I$)-LEN(X$)+1
4620  IF X$=I$[J1,J1+LEN(X$)-1] THEN 4650  Moving window match scan
4630  NEXT J1
4640  GØTØ 4540
4650  F=1                               Flag = "Y"
4660  RETURN
4680  DEF FNC(X)=INT(X/4.98)+1 Initializing stuff
4690  DEF FND(X)=X-INT((X-1)/4)*4
4700  Q$="*$ABCDEFGHIJKLMNØPQRSTUVWXYZ0987654321"   Legal character set
$710  DIM I$[72]
4720  DIM W$[60],X$[60],Y$[60],Z$[60]
4730  DATA 2,3,5,7,11,13,17,19,23,29 4730-4780 Put log (primes) into P array
4750  FØR J=1 TØ 10
4760  READ Q1
4770  P[J]=LØG(Q1)                      Line 4840 – Print "XXXX" into "E"    If you have:
4780  NEXT J                            1  T: HOW ARE YOU, $ NAM?
4790  M=L1=T1=T2=0                      2  A: $NAM
4800  T=0                               the "XXXX" will appear as an undefined $ variable
4810  MAT L=ZER                         value indicator
4812  DIM S[20]
4814  MAT S=ZER
4830  READ #2,1
4840  FØR J=1 TØ 5
4850  PRINT #2,J;"XXXXXX","XXXXXX","XXXXXX","XXXXXX"
$860  NEXT J
4870  RETURN
4890  DIM Q$[40]                        Converts D$ into log godelized hash code Q$
4900  DIM P[10]                         If illegal character, will stop ("_" is illegal)
4920  Q1=0
4930  J1=LEN(D$)
4940  IF J1<11 THEN 4960
4950  J1=10
4960  FØR J=1 TØ J1
4970  FØR K=1 TØ 36
4980  IF D$[J,J]=Q$[K,K] THEN 5020
4990  NEXT K
5000  NEXT J
5010  RETURN
5020  Q1=Q1+K*P[J]
5030  GØTØ 5000
5280  READ #1,FNB(B[I])+8               Get S$ from TEST
5290  FØR J=1 TØ FNA(B[I])
5300  READ #1;S$
5310  NEXT J
5320  IF I+1>240 THEN 5400
5330  IF A[I+1]#A[I] THEN 5400
5340  I6=B[I+1]
5350  READ #1,FNB(I6)+8
5360  FØR J=1 TØ FNA(I6)
5370  READ #1;A$
5380  NEXT J
5390  S$[LEN(S$)+1]=A$
5400  RETURN
5550  REM-DEBLANK S$                    Remove leading blanks from S$
5560  IF S$[1,1]#" " THEN 5600
5570  S$=S$[2]
5580  IF S$="" THEN 5600
5590  GØTØ 5560
5600  RETURN
6100  END
```

PAPER TAPES OF PILOT1 AND
PILOT2. ARE AVAILABLE FOR $10.00
FROM:

GREGORY YOB
2296 BRYANT PALO ALTO, CAL.
(415·326·4059)

USING PILOT

LOADING AND USING

If you are familiar with BASIC, the remarks will help you fit PILOT 1 and PILOT 2 to your system. If you aren't, follow this cookbook:

(1) Be sure your computer is an HP 2000 Series machine.

(2) LOGON, SCRATCH, ENTER THE CODE FOR PILOT 1 AND SAV AND LIS.

(3) Check that what you have is identical with this one. If not, fix and repeat this step.

(4) Repeat for PILOT 2.

(5) Create a dummy program, PILOT 3

```
5    COM K$(1)
10   PRINT "READ THE MANUAL, CHARLIE!"
15   K$ = "Y"
20   CHAIN – PILOT 1
30   END
```

Later you can write your own instructions.

(6) Now that you are loaded and ready (ahem):

```
OPEN–TEST,48
OPEN–E,5
GET–PILOT 1
RUN
```

At which time all those other errata will appear which you missed in Step 3.

It's wise to save some copies under other names in case you clobber this one [Beware of the CHAIN–BEAST]

(7) At last, it runs (sort of). Try the various operations – entering a program, LISTing, SCRatching, RENUMBERing, and RUNning.

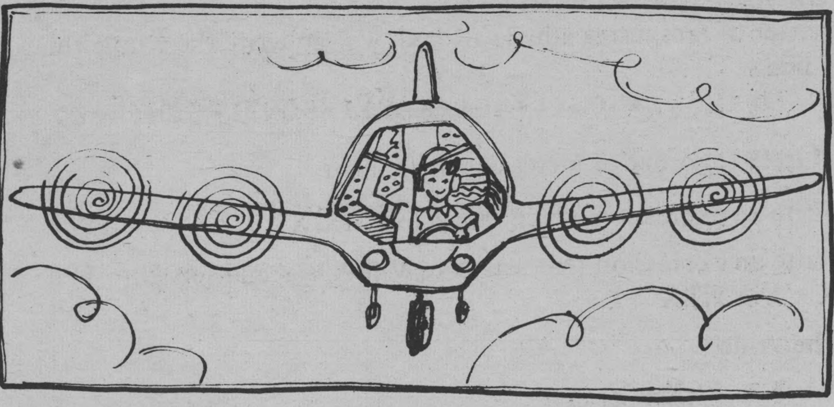
[When you RUN, PILOT 2 is activated. If you crash, you will still be in PILOT 2. When fixing your typos, be sure to get *SAV–THE PROGRAM YOU ARE FIXING* or you will likely either fix the wrong program or lose your fix when it chains to the other program] *Each time!!

(8) Try writing PILOT programs which use all the features (i.e., T Y N M J A R E).

(9) Congratulations!

Send us any really neat PILOT programs – some will appear in PCC.

(10) If you don't like steps 2 through 9, send $10 to Gregory Yob, 2296 Bryant, Palo Alto, CA. and we will send you a tape with PILOT 1, PILOT 2 and PILOT 3 on it.



MISCELLANEOUS THINGS & TECHNICAL HASSLES

(1) Files – If you have a highly similar BASIC (like NOVA or BASIC TYMSHARE), the files statement [Line 190] may be different. "TEST" is FILE #1 and "E" is FILE #2.

(2) Space – If you have a HP 2000C, your record length is 256 words so you can use twice as many strings, etc. per record. Try if you dare.

(3) Space (continued) – PILOT holds 240 single length lines. In practice about 1 in 8 lines are double length. Think of about 200 lines of PILOT as your maximum size.

(4) Since this is an interpreter, it looks at the disc a lot. There may be response problems at 5 or more terminals in PILOT. (This trouble has been experienced at SRI and LHS.) If you have this problem, let us know.

(5) There may still be bugs! Call me up (Greg at 415-326-4039) so we can fix em! If you have cleaner code or hot programming ideas, we have ears. (Especially if your version (a) works, (b) faster (c) with less core)

(6) Tapes of PILOT 1 and 2 are available – $10 service charge per tape (holds PILOT 1 and PILOT 2). When we have it together, manuals and some sample program will be included.

INTERPRETER vs. TRANSLATOR

IMPLEMENTING SIMPLE LANGUAGES ON MIDI-MINI TIMESHARE COMPUTERS

Pete Rowe
Lawrence Hall of Science
University of California
Berkeley

24 JAN 1973

In this and later issues of PCC, readers will be able to explore new languages. Some of the simple languages can be written in BASIC. The way they are implemented in BASIC is the point of this article.

At the start, I will define a Midi-Mini as a multi-user, BASIC interpreter with data files and string manipulation, timeshare computer. Hewlett Packard's 2000 series, Digital Equipment's EDU 30, 40, 50, RSTS-11 and Data General's Seminar 2 thru 10 generally comply with this definition.

BASIC as a problem oriented interpretive language is known for its ease of coding, debugging and editing. However, to use BASIC, one must have a minimal knowledge of algebra; a knowledge enjoyed by relatively few.

Then what other languages can we design for the majority of the "kids" (PCC vol.1,no.2 p.5) and teachers who do not and usually will not learn BASIC? PILOT, PYLON, CO-PILOT and NYLON are predecessors to PILOT 73. All are simple author languages that take only minutes to learn and use, but are powerful enough to produce moderately complex programs. The question arises of how to implement this new tool on currently available machines, providing the interactive features for PILCT 73 authors that BASIC authors have been enjoying for years.

Gear (1969) defines an interpreter as a routine that executes by statement-to-statement translation, substituting effective error tracing for execution efficiency. BASIC on all the afore mentioned computers has been implemented interpretively.

When PILOT-like languages are implemented interpretively in BASIC, also an interpreter, response-time is affected. PILOT 73 source code must be kept on a data file for interpretation, translation and/or editing. Hence the need for a Midi-Mini. And during file access and transfer, no machine instructions can be executed, hence reducing the number of machine instructions, therefore BASIC statements executed during a given amount of time. When many users are involved with file accesses and transfers then the number of instructions executed for an individual user will be even less, resulting in a degraded response-time.

Two actual examples are known: (1) A DIALOG program, a subset of a more elaborate interactive author language was implemented on our HP2000B as an interpreter and (2) Dr. Sylvan Rubin at S.R.I. implemented his PYLON interpretive language on the DEC RSTS-11. In both cases, seven terminals executing these interpreters seemed to be the magic number. The systems became bogged down doing file accesses and transfers and response-time was noticeably degraded.

An operational solution: Create an editor, syntax analyzer and a PILOT 73-to-BASIC translator. The editor and statement syntax analyzer could reside in one BASIC program, where each PILOT 73 line, before insertion to a source file, would be checked for acceptable PILOT 73 grammar. Once insertion and editing were complete, the author could give a command to translate his PILOT 73 statements into BASIC code, which would be written into a data file. This translation need only be done once to produce the executable BASIC code. On our Decision system and on the Data General Seminar series, a user could access this data file as if it were a program file and execute it using the machine's BASIC interpreter. On the HP2000 series, an intermediate step of punching a paper tape image of the data file and loading the tape back into the terminal as a BASIC program, is necessary.

Perhaps in a student-as-author environment, handling paper tape will discourage their involvement. It's yet unknown what effect the intermediate step might have on teacher authors.

In contrast to the interpretive implementation of PILOT 73, a translator need only translate the source code once. And in effect, the PILOT 73 author is creating a BASIC program, eliminating file accesses during its execution and therefore improving response-time.

PILOT 73 is a real language! The following people have agreed on the main features of PILOT (this version is a subset) and the means of extension. Specifications are available through U.C. Medical Center.

WHO'S WHO IN PILOT LAND

Name	Organization	Mainframe	Language
John Starkweather	*UC Medical Center*	*IBM 360*	*PL/I*
Marty Kamp	*San Francisco*	*Datapoint 2200*	*Machine Code*
Pete Rowe	*LHS*	*Decision*	*BASIC*
	Berkeley	*HP 3000*	*BASIC*
Dean Brown	*Stanford Research*	*Tymshare*	*BASIC*
Sylvan Rubin	*Institute*	*DEC11/20* *RSTS*	*SUPER BASIC*
Gregory Yob	*PCC*	*HP2000C,E,F*	*BASIC*

If you have a version, let us know and we will add you to the list.

Dean Brown, Marty Kamp, and Greg Yob are interested in groovy programs, curricula, etc., which develop in PILOT.

THE PROGRAMMER'S TOOLBOX

by marc le brun

[Each issue we will present an "advanced" programming technique: with explanations, examples, programs and problems. We welcome suggestions for topics of interest to you.]

LOGICAL EXPRESSIONS

A logical expression is an expression whose value is either 0 or 1 depending on the values of the variables in the expression. For example

ABS(SGN(X))

is a logical expression

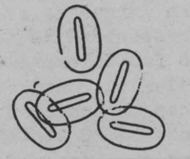
IF X = 0 THEN the expression equals 0
IF X ≠ 0 THEN the expression equals 1

Notice the use of the words IF and THEN. It is sometimes convenient to think of 1 as representing *true* and 0 as representing *false.*

Many times it is possible to use a logical expression in place of an IF-THEN statement. In this article we will show how this is done.

In the following discussion we will use an e to represent a logical expression, and other lower case letters to represent any old kind of expression.

Notice the following "rules."



IF	THEN
$e = 0$:	$1 - e = 1$
$e = 1$:	$1 - e = 0$

$1 - e$ is always the "opposite" of e. If we think of e as being equivalent to TRUE or FALSE then $1 - e$ is equivalent to NOT e; that is, NOT TRUE (FALSE) and NOT FALSE (TRUE).

Remember also that any number multiplied by 0 is 0, and that any number plus 0 is that number.

Now suppose we wish to write an expression which is equal to a if e is one and equal to b if e is zero. Here is how we do it:

$a*e + b*(1 - e)$

Suppose we wish to set X to the value of this expression. The following two BASIC programs do this in different ways

PROGRAM 1

```
10 LET X=A*E+B*(1-E)
20 ...
```

PROGRAM 2

```
10 IF E=1 THEN 40
20 LET X=B
30 GO TO 50
40 LET X=A
50 ...
```

Clearly Program No. 1 is a lot simpler.

If we want a variable to have a certain value if something is true and another value if it is false, then in MOST cases we can use a logical expression instead of an IF-THEN statement.

The rest of this article will be devoted to showing how to write logical expressions for the usual sorts of "if's." First we will consider what are called "relational expressions," that is, those involving =, < , >, <=, >= and <> (or #). Here is a table giving the relational expressions and their equivalent logical expressions.

RELATIONAL EXPRESSION	LOGICAL EXPRESSION
$p = q$	1 – ABS(SGN($p-q$))
$p < q$	1 – SGN(1 + SGN($p-q$))
$p > q$	1 – SGN(1 – SGN(p-q))
$p <= q$	SGN(1 – SGN($p-q$))
$p >= q$	SGN(1 + SGN($p-q$))
$p <> q$	ABS(SGN($p-q$))

For example, the phrase

"IF $p <> q$ . . . "

is equivalent to

"IF ABS(SGN($p-q$)) = 1 . . . "

Secondly, we will consider what are called the "logical operators," that is, those involving AND, OR, NOT and others.

We already know that NOT e is equivalent to $1 - e$. If we have two logical expressions $e1$ and $e2$ then $e1$ AND $e2$ is equivalent to $e1*e2$. $e1$ OR $e2$ is $e1 + e2$. (Remember the two facts about zero given above.)

here theY are ;

SHORT FORM	LONG FORM	LOGICAL EXPRESSION
e1 AND e2	e1 AND e2	e1*e2
e1 OR e2	NOT(NOT(e1) AND NOT (e2))	1 – (1–e1)*(1–e2)
e1 XOR e2	(e1 AND NOT(e2)) OR (NOT(e1) AND e2)	e1*(1–e2)+(1–e1)*e2
e1 NAND e2	NOT(e1 AND e2)	1 – e1*e2
e1 NOR e2	NOT(e1) AND NOT(e2)	(1–e1)*(1–e2)
e1 EQV e1	(e1 AND e2) OR (NOT(e1) AND NOT (e2))	e1*e2 + (1–e1)*(1–e2)
e1 IMP e2	NOT(e1 AND NOT(e2))	1 – e1*(1–e2)
e1 NIMP e2	e1 AND NOT (e2)	e1*(1–e2)

For example the phrase

IF a OR b . . .

is equivalent to

IF 1 – (1–$e1$)*(1–$e2$) = 1 . . .

In many cases the resulting expressions can be algebraically simplified. It is also useful to use several LET statements rather than repeating the same sub-expression. An example – suppose we wish to set U to V if X >= Y and X <= Z; and to set U to W if X <Y or X > Z. Here is a short program which does this:

```
    10 LET T=SGN(1+SGN(X+Y))*SGN(1-(X-Z))
    20 LET U=V*T+W*(1-T)
```

As a matter of fact, using a little algebra, we can write the whole thing in one line –

```
10 LET U=(V-W)*SGN(1+SGN(X-Y))*SGN(1-SGN(X-Z))+W
```

PROBLEMS (in order of increasing difficulty)

1. Write an expression which sets Z to MAX(X,Y).
2. Write an expression that sets U to V if X < Y and sets U to W if X < Y AND X < Z.
3. The "rule"

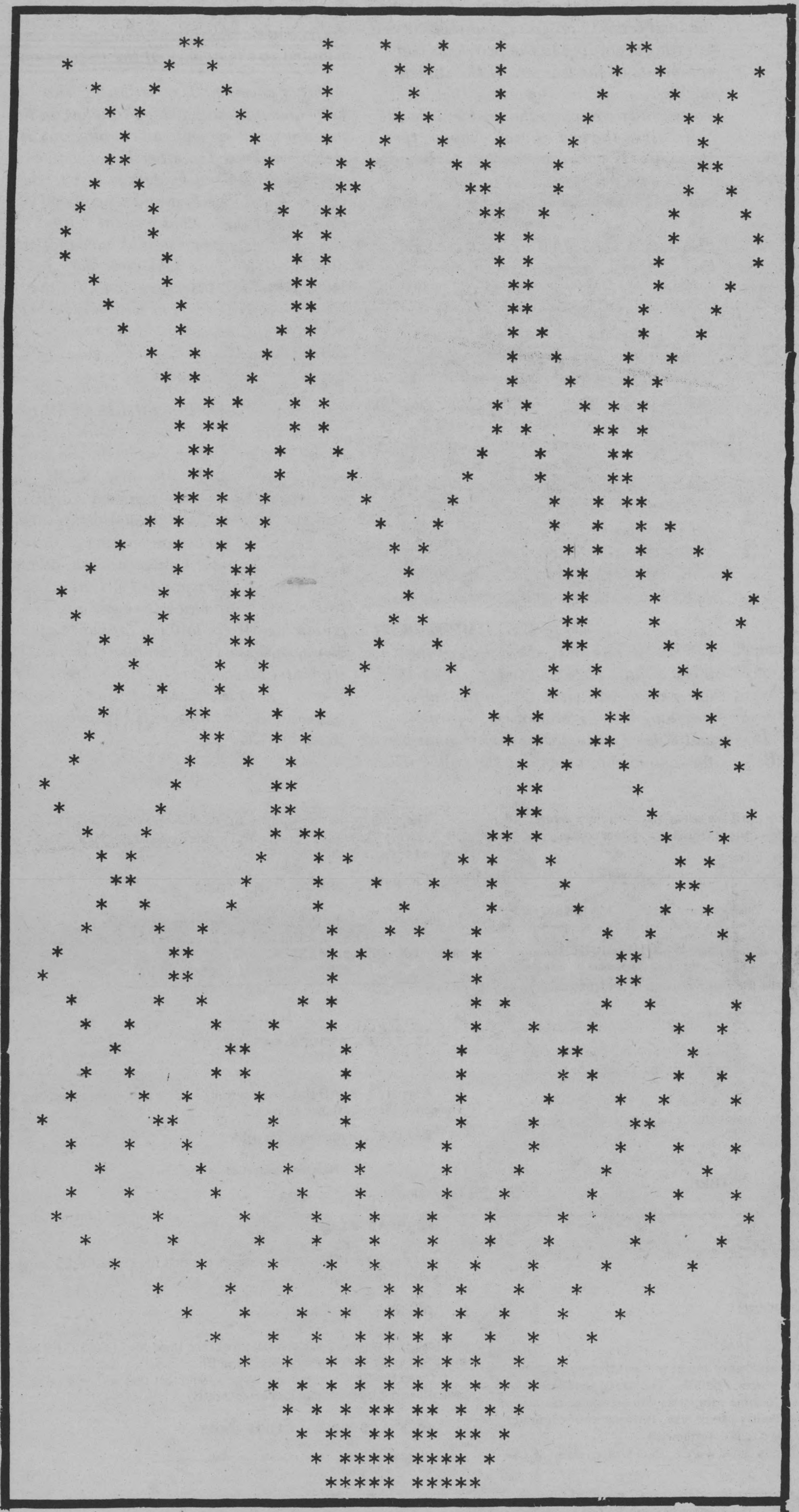
   X = Y : 1/(X + Y)
   X <> Y : 1/(X – Y)

   has to be done with IF-THEN – why? Think up some more "rules" which can't be performed with a logical expression. Can you find a way to describe when you have to use IF-THEN?
4. *Without* using the MOD operator, write an expression which is TRUE if an *odd* number of *e*'s in the set {*e1, e2, e3*} are TRUE and FALSE if an *even* number are TRUE.

   Hint: Explore the properties of XOR.
5. *Logical* expressions can only "select" one of two values. Write an "illogical" expression which "selects" one of three values according to the following rules.

   X > Y : a
   X = Y : b Hint: Think about quadratics
   X < Y : c
6. Can you think of a general class of "selection" expressions which picks one of N values? Describe this class. Hint: Think about Problem 5.
7. Can you generalize the logical operators to get some operators which "make sense" to use in combining "selection" expressions? Hint: Think about matrices

DO - IT - YOURSELF !



HERE ARE SOME EXAMPLES OF PICTURES WE MADE.

NOW IT'S YOUR TURN:

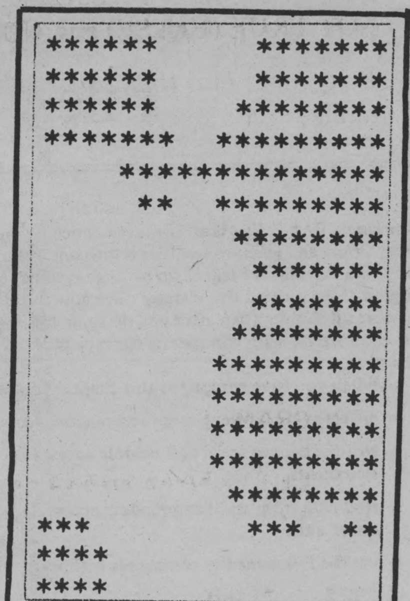
1. FIND THE HURKLE AND DRAW A PICTURE OF IT. (SEE p.22)
2. WRITE A PROGRAM WHICH DRAWS A DIFFERENT PICTURE EVERY RUN.

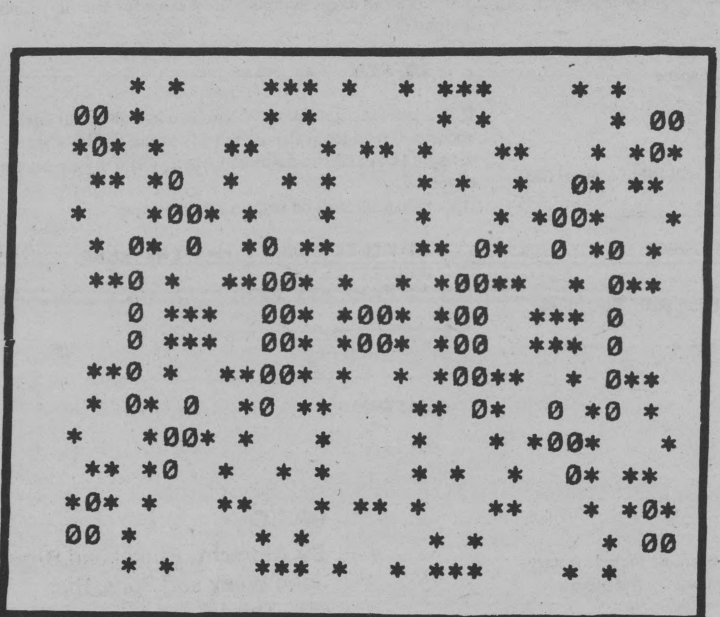
WHICHEVER YOU DO, SEND THE RESULT TO US. IF WE LIKE IT WE WILL:

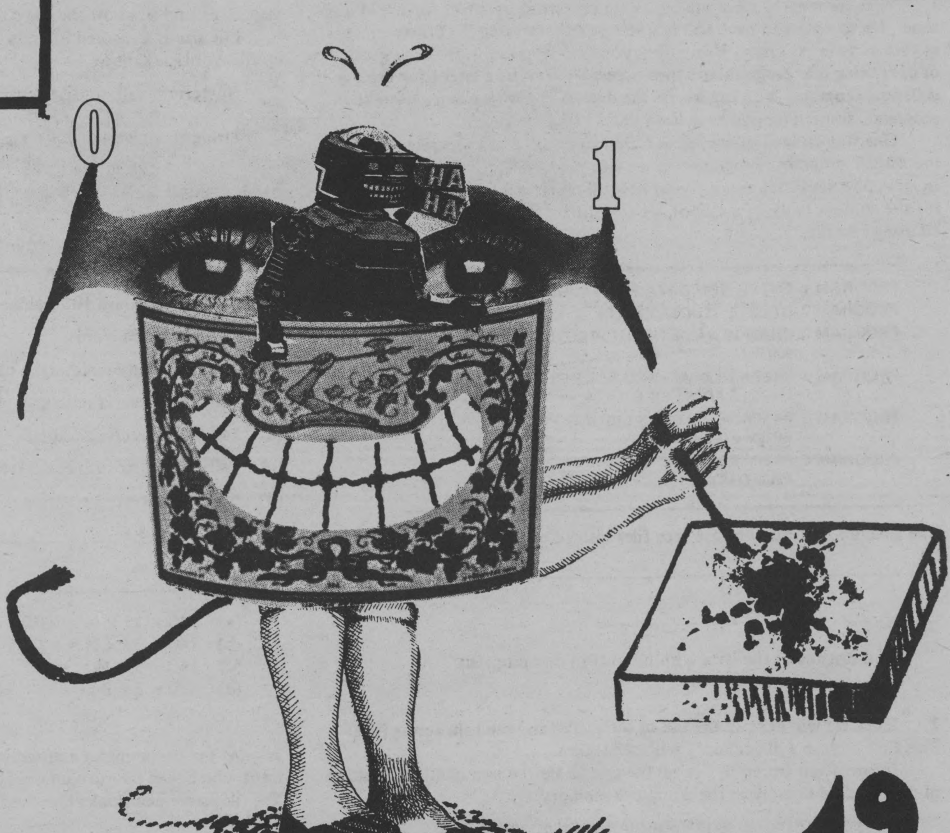
1. PRINT IT IN THE NEXT ISSUE OF P.C.C.
2. SEND YOU EXTRA COPIES TO SHOW YOUR FRIENDS.
3. GIVE YOU A FREE SUBSCRIPTION TO P.C.C.
4. MAIL YOU A SPECIAL SURPRIE !!

«Dragonflies are the machines of the future.»

Salvador Dali







I suppose it is somewhat gauche to review your own book, but we worked hard on it and I'm reasonably satisfied with the result. As an instructional text, it works, and I judge it to be among the best of its kind now available, perhaps even the best (but consider the source . . .).

In contrast to our instructional workbook MY COMPUTER LIKES ME, the Wiley BASIC book is in programmed instruction format. Programmed instruction, if done properly, entails a process of careful analysis and sequencing of the material, vocabulary, and concepts to be taught, in order to attain explicit learning objectives stated out front. That means the student doesn't have to guess about what he or she is supposed to learn and be tested on. The objectives specify particular observable behaviors which the learner must be able to demonstrate after having worked through the instructional program in the manner specified. Here is what we specified

With the programmed instruction format, you'll be actively involved in learning BASIC. The material is presented in short numbered sections called frames, each of which gives you a question or asks you to write a program. Correct answers are given below the dashed line. For the best results, we urge you to take pen or pencil in hand and to use a piece of thick paper or cardboard to keep the answers out of sight until you have written your answer in the space provided. The questions are carefully designed to call your attention to important points in the examples and explanations, and to help you learn to apply what is being explained or demonstrated.

Now, understand that this book is one of an extensive series of self-instructional books that Wiley publishes. The series is principally marketed through college bookstores on special racks containing all the Wiley self-teaching books. Given the amount of depersonalized mass instruction and the number of harrassed and/or disinterested and/or incompetent instructors in the typical college, Wiley sees potential profit in entering the "college outline" publishing competition. Students unlucky enough to receive poor instruction, or indolent to the point of needing last minute remedial instruction, or perceptive, energetic and ambitious enough to seek additional sources of learning will often buy this style of self-study text.

But for us (the authors), the self-instructional aspect was a problem, because we believe in learning by doing. In contrast, the Wiley BASIC text was to be self-contained, meaning that we were not to assume that the reader would have any access to a computer terminal – which in our experience is where the real learning action takes place. To the end of the introductory section where we encourage the reader to get access to a terminal for practice, Wiley's editors added: "However, computer access is not essential, all you need is this Self-Teaching Guide."

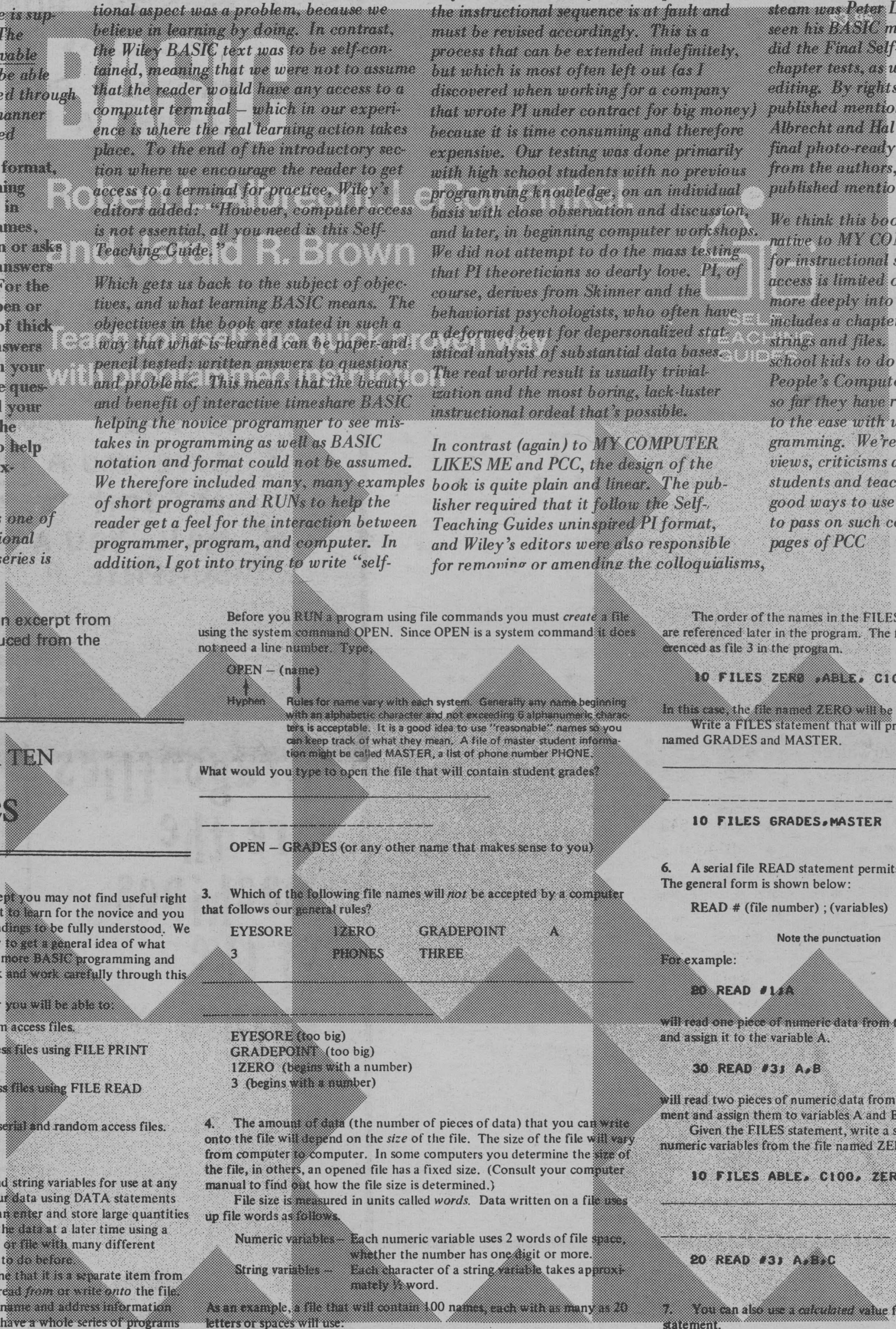
Which gets us back to the subject of objectives, and what learning BASIC means. The objectives in the book are stated in such a way that what is learned can be paper-and-pencil tested: written answers to questions and problems. This means that the beauty and benefit of interactive timeshare BASIC helping the novice programmer to see mistakes in programming as well as BASIC notation and format could not be assumed. We therefore included many, many examples of short programs and RUNs to help the reader get a feel for the interaction between programmer, program, and computer. In addition, I got into trying to write "self-explaining" demonstration programs that showed or explained how they worked or what they did when RUN. But in some cases, a lot of time and space had to be used to explain and teach that which would have been almost trivial if we could have assumed that the reader were seated at a TTY or CRT.

Another aspect of programmed instruction as a technique is the requirement of testing the instructional program, and revising and retesting according to the problems and comments of the learner. If the student is unable to meet the objectives, that is, successfully complete the problems in the Self-Test at the end of each chapter, then the instructional sequence is at fault and must be revised accordingly. This is a process that can be extended indefinitely, but which is most often left out (as I discovered when working for a company that wrote PI under contract for big money) because it is time consuming and therefore expensive. Our testing was done primarily with high school students with no previous programming knowledge, on an individual basis with close observation and discussion, and later, in beginning computer workshops. We did not attempt to do the mass testing that PI theoreticians so dearly love. PI, of course, derives from Skinner and the behaviorist psychologists, who often have a deformed bent for depersonalized statistical analysis of substantial data bases. The real world result is usually trivialization and the most boring, lack-luster instructional ordeal that's possible.

In contrast (again) to MY COMPUTER LIKES ME and PCC, the design of the book is quite plain and linear. The publisher required that it follow the Self-Teaching Guides uninspired PI format, and Wiley's editors were also responsible for removing or amending the colloquialisms, conversational asides, and ironic humor natural to the authors' collective writing style. I finally got the feeling that the publishers wished the book to have Lowest Common Denominator appeal, like commercial TV, on the theory that it would broaden the sales potential. However, to give credit where credit is certainly due, Wiley editor Irene Brownstone provided us with exceptionally helpful suggestions and excellent detailed critiques of the various manuscript drafts and revisions, which definitely contributed to the quality of the instruction.

Another anonymous contributor who helped us put the finishing touches on the manuscript as we were all running out of steam was Peter Lynn Sessions (you've seen his BASIC music ideas in PCC). He did the Final Self-Test and several end of chapter tests, as well as general helpful editing. By rights he should have received published mention. Likewise, Mary Jo Albrecht and Hal Hershey, who did the final photo-ready layout with a little help from the authors, should have received published mention.

We think this book is an excellent alternative to MY COMPUTER LIKES ME for instructional situations where terminal access is limited or sporadic. It also goes more deeply into BASIC than MCLM, and includes a chapter on Advanced BASIC and strings and files. I assign chapters to high school kids to do before coming to the People's Computer Center workshops, and so far they have responded enthusiastically to the ease with which they get into programming. We're looking forward to reviews, criticisms and comments from students and teachers on the content and good ways to use this book, and we hope to pass on such comments through the pages of PCC


BASIC
Robert L. Albrecht, LeRoy Finkel,
and Jerald R. Brown
Teach yourself the quick, proven way
with programmed instruction
SELF-TEACHING GUIDES

The following material is an excerpt from Chapter 10 of BASIC, reduced from the actual 7" x 10" page size.

CHAPTER TEN

Files

The use of BASIC files is an advanced concept you may not find useful right away. *How* and *when* to use files is difficult to learn for the novice and you may find this chapter takes two or three readings to be fully understood. We suggest that you read this chapter once now to get a general idea of what files are all about. Then after you do some more BASIC programming and are comfortable with computers, come back and work carefully through this chapter.

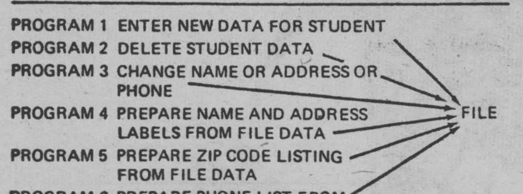
When you have completed this chapter you will be able to:

- differentiate between serial and random access files.
- write data onto serial and random access files using FILE PRINT statements.
- read data from serial and random access files using FILE READ statements.
- use the following file commands with serial and random access files.

FILES IF END TYP

1. Files are used to store numeric data and string variables for use at any time. Up to now you have had to enter your data using DATA statements as part of your program. Using files, you can enter and store large quantities of data using one program and then access the data at a later time using a *different* program. You can access the data or file with many different programs, something you have been unable to do before.

One way to look at the file is to imagine that it is a separate item from the BASIC program. Programs are used to read *from* or write *onto* the file. In an application that uses a file to hold all name and address information for the student body of a school, we might have a whole series of programs all using *one* file.


PROGRAM 1 ENTER NEW DATA FOR STUDENT
PROGRAM 2 DELETE STUDENT DATA
PROGRAM 3 CHANGE NAME OR ADDRESS OR PHONE
PROGRAM 4 PREPARE NAME AND ADDRESS LABELS FROM FILE DATA
PROGRAM 5 PREPARE ZIP CODE LISTING FROM FILE DATA
PROGRAM 6 PREPARE PHONE LIST FROM FILE DATA
FILE

One advantage of placing data into files instead of using DATA statements is

____

- - - - - - - - - - - - - - - -

You can access the data with more than one program.

2. Later we will explain the use of serial files and random access files. This first section will deal only with serial files.

Information stored in a serial file can be viewed as a continuous series of data packed densely in the computer memory.

GEORGE/YOUNG/25/94191/BOB/HARRIS/42/83107/ . . .

To get to data in the middle of a serial file you must read from the beginning of the file, one piece of data at a time, until you reach the data you need.

Before you RUN a program using file commands you must *create* a file using the system command OPEN. Since OPEN is a system command it does not need a line number. Type,

OPEN – (name)

Hyphen

Rules for name vary with each system. Generally any name beginning with an alphabetic character and not exceeding 6 alphanumeric characters is acceptable. It is a good idea to use "reasonable" names so you can keep track of what they mean. A file of master student information might be called MASTER, a list of phone number PHONE.

What would you type to open the file that will contain student grades?

____

- - - - - - - - - - - - - - - -

OPEN – GRADES (or any other name that makes sense to you)

3. Which of the following file names will *not* be accepted by a computer that follows our general rules?

EYESORE	1ZERO	GRADEPOINT	A
3	PHONES	THREE	

____

- - - - - - - - - - - - - - - -

EYESORE (too big)
GRADEPOINT (too big)
1ZERO (begins with a number)
3 (begins with a number)

4. The amount of data (the number of pieces of data) that you can write onto the file will depend on the *size* of the file. The size of the file will vary from computer to computer. In some computers you determine the size of the file, in others, an opened file has a fixed size. (Consult your computer manual to find out how the file size is determined.)

File size is measured in units called *words*. Data written on a file uses up file words as follows.

Numeric variables – Each numeric variable uses 2 words of file space, whether the number has one digit or more.

String variables – Each character of a string variable takes approximately ½ word.

As an example, a file that will contain 100 names, each with as many as 20 letters or spaces will use:

String variable = 100 × 20 = 2000 characters
= 2000 × ½ = 1000 words of file space

A file that will contain 100 numbers will use:

100 × 2 = 200 words

Calculate how many words each of these sets of data will fill in a serial file.

(a) 140, 15-character names ______________
(b) 140, 20-character addresses ______________
(c) 140, 5-character zip codes (string variable) ______________
(d) 420 numbers (representing responses to an opinion poll. Responses are 1, 2, or 3.) ______________

- - - - - - - - - - - - - - - -

(a) 140 × 15 × ½ = 1050
(b) 140 × 20 × ½ = 1400
(c) 140 × 5 × ½ = 350
(d) 420 × 2 = 840

5. At the beginning of a program that uses files you must include a statement which tells the computer which files are to be used by the program. The files statement looks like this:

```
10 FILES ABLE, C100, ZERØ
           1     2     3
```

The order of the names in the FILES statement determines how they are referenced later in the program. The file named ZERO will now be referenced as file 3 in the program.

```
10 FILES ZERØ ,ABLE, C100
```

In this case, the file named ZERO will be referenced as file 1.

Write a FILES statement that will prepare the computer to use files named GRADES and MASTER.

____

- - - - - - - - - - - - - - - -

```
10 FILES GRADES,MASTER
```

6. A serial file READ statement permits reading data from an existing file. The general form is shown below:

READ # (file number) ; (variables)

Note the punctuation

For example:

```
20 READ #1;A
```

will read one piece of numeric data from the first file in the FILES statement and assign it to the variable A.

```
30 READ #3; A,B
```

will read two pieces of numeric data from the third file in the FILES statement and assign them to variables A and B.

Given the FILES statement, write a statement that will read three numeric variables from the file named ZERO.

```
10 FILES ABLE, C100, ZERØ
```

____

- - - - - - - - - - - - - - - -

```
20 READ #3; A,B,C
```

7. You can also use a *calculated* value for the file number in a file READ statement.

```
20 READ #X; A$,B
```

If in a previous statement, X has been calculated as equal to 2, the statement above will read from the second file in the FILES statement. The string variable (A$) and the numeric variable (B) will be read each time Line 20 is executed.

Which file will be read in the following:

```
10 FILES PHØNE, MASTER, ZERØ
20 LET Y=3-1
30 READ #Y; A$,B
```

____

- - - - - - - - - - - - - - - -

MASTER

BASIC
By Albrecht, Finkel and Brown
John Wiley and Sons, Inc.
605 Third Avenue
New York, N.Y. 10016
325 pages, 1973.

HOW TO BUY AN EDUSYSTEM

O.K. Here is how you build an Edu 10 or a *one user* Edu 20.

Edu 10		One User Edu 20	
• PDP8E–BA	$4490	• PDP8E–BE	$5650
• MI8–EF	500	• MI8–EF	500
• LT33–DC	1620	• KP8–E	250
• Software	250	• LT33–DC	1620
• Textbook Kit	100	• Edu 20 Software	250
	$6960	• Textbook Kit	100
			$8370

Now here are four ways to get a 4 TTY Edu 20 with 8K memory.

(1) Buy an Edu 10 this year and next year expand to an 8K Edu 20 with 4 TTYs.

• Edu 10	6960	
• MC8–E + installation	2750 +	150
• KP8–E + installation	250 +	60
• 3 LT33-DC + installation	4860 +	360
• 3 KL8-E + installation	900 +	180
• Edu 20 software	250	
	15970	750
Total cost	$16720	

(2) Buy a one user Edu 20 with 8K this year – next year expand to 4 TTY's.

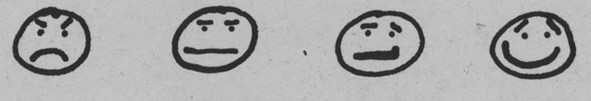
• One user Edu 20 with 8K	8370	
• 3 LT33-DC + installation	4860 +	360
• 3 KL8-E + installation	900 +	180
	14130	540
Total cost	$14670	

(3) Buy a 4 user Edu 20 with 8K *this* year.

• Edu 20 with 8K and 1 TTY	8370
• 3 LT33–DC	4860
• 3 KL8–E	900
Total cost	$14130

(4) Buy an 8K Edu 20 with one TTY from DEC and buy 3 TTYs from someone else.*

• Edu 20 with 8K and 1 TTY	8370
• 3 TTYs from someone else*	3450
• 3 KL8–E from DEC	900
Total cost	$12720



$16720 $14670 $14130 $12720

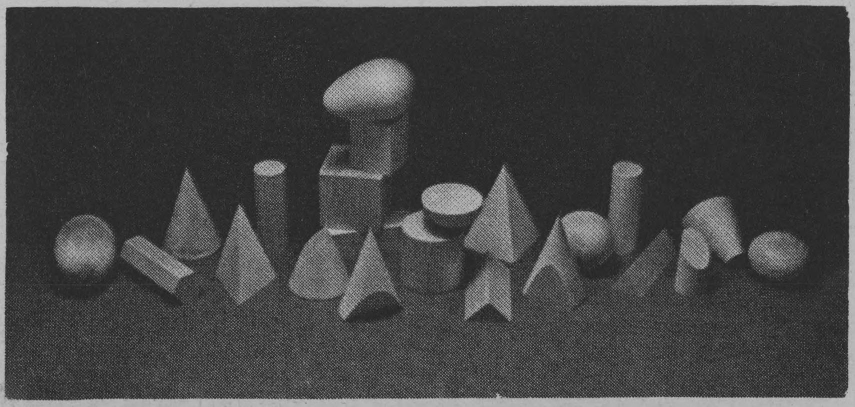
**We got thisprice from Data Terminals Corporation. $1150 for a new ASR 33 TTY modified so taht it will work on a PDP8E. Or you can get a rebuilt ASR 33 for PDP8 from DTC for $850. For more information, contact Data Terminals Corporation, P.O. Box 5583, San Jose, CA. 95150. Phone (408)378-1112.*

Power fail detect and restart is handy! It keeps your software from getting wiped out during power failures or temporary brownouts or when someone trips over the power cord. You just restart when power is OK again – otherwise, you usually have to reload the software.

Last time I talked about Edu 10 and Edu 20 and how to get from Edu 10 to Edu 20 and the cost of adding TTYs and stuff like that. This time I'll start by giving you a handy do-it-yourself kit consisting of excerpts from DEC's price list.

ID #	DESCRIPTION	PRICE	INST* CHG	MONTHLY MAINTENANCE
PDP8E–BA	Computer, 4K memory, TTY control	4490	––	60
PDP8E–BE	Computer, 8K memory, TTY control	5650	––	80
MC8–E	4K memory. Required to expand from 4K to 8K	2750	150	20
MM8–E	4K memory. This one gets you from 8K to 12K or from 12K to 16K	2500	150	20
MC8–EJ	8K memory. Required to expand from 4K to 12K	4150	175	40
MM8–EJ	8K memory. Gets you from 8K to 16K in one neat jump	3900	175	40
KP8–E	Power fail/restart	250	60	2
MI8–EF	Hardware bootstrap	500	60	5
LT33–DC	Teletype Model ASR 33 modified to work with PDP8	1620	120	30
KL8–E	Interface card for TTY	300	60	10

**This is the charge for installing additional equipment on an existing system.*



Next – let's look at more memory. First, suppose we buy a 16K Edu 20 with one TTY.

• PDP8E–EJ	5650
• MM8–EJ	3900
• Hardware bootstrap	500
• Power fail/restart	250
• LT33-DC	1620
• Edu 20 software	250
• Textbook kit	100
Total cost	$12270

You can now add on TTYs – up to 8 of them with the 16K version of Edu 20. Or you may prefer using Edu 21 software which provides modest *string* capabilities.

We will, in turn, look at another way to get a 16K Edu 20 or Edu 21 with one TTY.

• Start with Edu 10	6960		4K
• Add on things at later times			
KP8–E + installation	250 +	60	
MC8–E + installation	2750 +	150	8K
MM8–E + installation	2500 +	150	12K
MM8–E + installation	2500 +	150	16K
Edu 20 or Edu 21 software	250		
	15210 +	510	
Total cost	$15720		

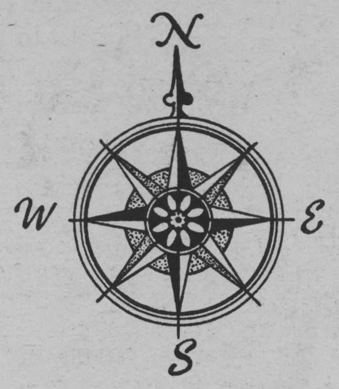
I'll leave other possibilities to you – like going from 4K to 12K then to 16K or from 4K to 8K then to 16K and so on.

In the meantime, I'll rest up for the next issue of PCC when I'll talk about EduSystem 25 and maintenance and . . . what do *you* want to know? Write a letter!

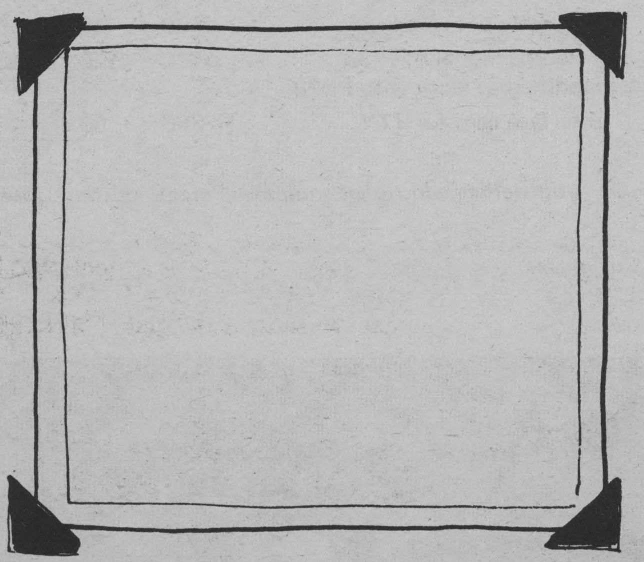
BA

THE HURKLE IS A HAPPY BEAST

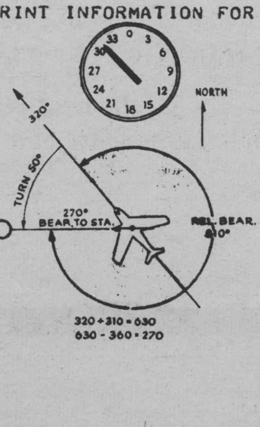
Hurkle? A Hurkle is a happy beast and lives in another galazy on a planet named Lirht that has three moons. Hurkle are favorite pets of the gwik, the dominant race of Lirth and . . . well, to find out more, read "The Hurkle is a Happy Beast" in a book called A WAY HOME by Theodore Sturgeon, published by Pyramid Publications, 444 Madison Avenue, New York, NY 10022. (Unless they have moved since January, 1968.)



Happy hurkles radiate.
Scared hurkles go invisible.
We took a snapshot of a happy radiating hurkle, but the camera click scared him and . . . .



```
100  REM *** HURKLE - PEOPLE'S COMPUTER COMPANY, MENLO PARK, CA
110  RANDOM
120  REM *** N IS THE NUMBER OF GUESSES ALLOWED
130  LET N=5
140  PRINT "DO YOU WANT THE RULES (1=YES  0=NO)";
150  INPUT Z
160  IF Z <> 1 THEN 490
170  REM *** HERE ARE THE RULES
180  PRINT "A HURKLE IS HIDING IN A GRID, LIKE THE ONE BELOW."
190  PRINT
200  PRINT
210  PRINT TAB(26);"NORTH"
220  PRINT
230  FOR K=9 TO 0 STEP -1
240  IF K <> 4 THEN 270
250  PRINT TAB(8);"WEST   4";TAB(20);". . . . . . . . . .   EAST"
260  GOTO 280
270  PRINT TAB(14);K;TAB(20);". . . . . . . . . ."
280  NEXT K
290  PRINT
300  PRINT TAB(20)"0 1 2 3 4 5 6 7 8 9"
310  PRINT
320  PRINT TAB(26);"SOUTH"
330  PRINT
340  PRINT "TRY TO GUESS WHERE THE HURKLE IS HIDING. YOU GUESS"
350  PRINT "BY TELLING ME THE GRIDPOINT WHERE YOU THINK THAT"
360  PRINT "THE HURKLE IS HIDING. HOMEBASE IS POINT  0,0  IN"
370  PRINT "THE SOUTHWEST CORNER. YOUR GUESS SHOULD BE A PAIR"
380  PRINT "OF WHOLE NUMBERS, SEPARATED BY A COMMA. THE FIRST"
390  PRINT "NUMBER TELLS HOW FAR TO THE RIGHT OF HOMEBASE AND"
400  PRINT "THE SECOND NUMBER TELLS HOW FAR ABOVE HOMEBASE YOU"
410  PRINT "THINK THE HURKLE IS HIDING. FOR EXAMPLE, IF YOU "
420  PRINT "THINK THE HURKLE IS 7 TO THE RIGHT AND 5 ABOVE"
430  PRINT "HOMEBASE, YOU ENTER  7,5  AS YOUR GUESS AND THEN"
440  PRINT "PRESS THE 'RETURN' KEY. AFTER EACH GUESS, I WILL"
450  PRINT "TELL YOU THE APPROXIMATE DIRECTION TO GO FOR YOUR"
460  PRINT "NEXT GUESS. GOOD LUCK!"
470  PRINT
480  REM *** HURKLE 'PICKS' A GRIDPOINT AND HIDES
490  LET A=INT(10*RND(0))
500  LET B=INT(10*RND(0))
510  PRINT
520  PRINT "THE HURKLE IS HIDING - YOU GET";N;"GUESSES TO FIND HIM."
530  PRINT
540  REM *** GET A GUESS AND PRINT INFO FOR PLAYER
550  FOR K=1 TO N
560  PRINT "WHAT IS YOUR GUESS";
570  INPUT X,Y
580  IF ABS(X-A)+ABS(Y-B)=0 THEN 710
590  REM *** GO TO INFO SUBROUTINE
600  GOSUB 760
610  PRINT
620  NEXT K
630  PRINT
640  REM *** HURKLE WAS NOT FOUND IN N GUESSES
650  PRINT "SORRY, THAT'S";N;"GUESSES."
660  PRINT "THE HURKLE IS AT ";A;",";B
670  PRINT
680  PRINT "LET'S PLAY AGAIN."
690  GOTO 490
700  REM *** HURKLE HAS BEEN FOUND!
710  PRINT
720  PRINT "YOU FOUND HIM IN"K;"GUESSES!!!"
730  PRINT "LET'S PLAY AGAIN."
740  GOTO 490
750  REM *** SUBROUTINE: PRINT INFORMATION FOR NEXT GUESS
760  PRINT "GO ";
770  IF Y=B THEN 820
780  IF Y<B THEN 810
790  PRINT "SOUTH";
800  GOTO 820
810  PRINT "NORTH";
820  IF X=A THEN 870
830  IF X<A THEN 860
840  PRINT "WEST";
850  GOTO 870
860  PRINT "EAST";
870  PRINT
880  RETURN
890  END
```

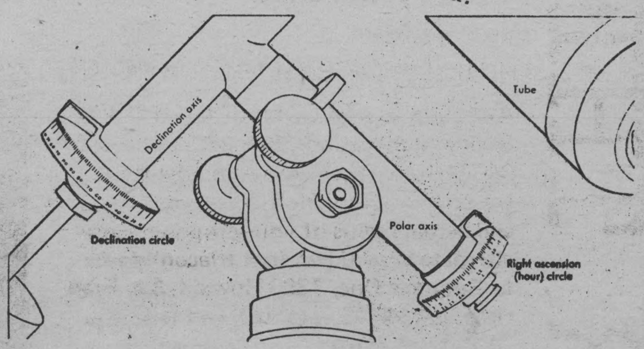


```
RUN
DO YOU WANT THE RULES (1=YES  0=NO)? 1
A HURKLE IS HIDING IN A GRID, LIKE THE ONE BELOW.

                         NORTH

             9     . . . . . . . . . .
             8     . . . . . . . . . .
             7     . . . . . . . . . .
             6     . . . . . . . . . .
             5     . . . . . . . . . .
       WEST  4     . . . . . . . . . .   EAST
             3     . . . . . . . . . .
             2     . . . . . . . . . .
             1     . . . . . . . . . .
             0     . . . . . . . . . .

                   0 1 2 3 4 5 6 7 8 9

                        SOUTH

TRY TO GUESS WHERE THE HURKLE IS HIDING. YOU GUESS
BY TELLING ME THE GRIDPOINT WHERE YOU THINK THAT
THE HURKLE IS HIDING. HOMEBASE IS POINT  0,0  IN
THE SOUTHWEST CORNER. YOUR GUESS SHOULD BE A PAIR
OF WHOLE NUMBERS, SEPARATED BY A COMMA. THE FIRST
NUMBER TELLS HOW FAR TO THE RIGHT OF HOMEBASE AND
THE SECOND NUMBER TELLS HOW FAR ABOVE HOMEBASE YOU
THINK THE HURKLE IS HIDING. FOR EXAMPLE, IF YOU
THINK THE HURKLE IS 7 TO THE RIGHT AND 5 ABOVE
HOMEBASE, YOU ENTER  7,5  AS YOUR GUESS AND THEN
PRESS THE 'RETURN' KEY. AFTER EACH GUESS, I WILL
TELL YOU THE APPROXIMATE DIRECTION TO GO FOR YOUR
NEXT GUESS. GOOD LUCK!


THE HURKLE IS HIDING - YOU GET 5 GUESSES TO FIND HIM.

WHAT IS YOUR GUESS? 5,5
GO NORTHWEST

WHAT IS YOUR GUESS? 3,7
GO SOUTH

WHAT IS YOUR GUESS? 3,6

YOU FOUND HIM IN 3 GUESSES!!!
LET'S PLAY AGAIN.

THE HURKLE IS HIDING - YOU GET 5 GUESSES TO FIND HIM.

WHAT IS YOUR GUESS?
```

This is grid point 7,5

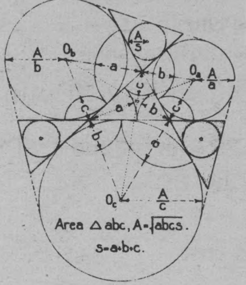
Lucky!

I'll quit while I'm ahead.



To change number of guesses allowed, change Line 130.

Lines 210 – 320 print a 10 by 10 grid (see RUN below). How would you change the program to print a larger or smaller grid – or a grid of size G by G?



To RUN on HP 2000, delete Line 110.

Change the game –

- First number is distance above and second number is distance to the right of homebase
- Longitude and latitude?
- Number the grid in rows and columns, like this

```
                         NORTH

             1  2  3  4  5  6  7  8  9  10

        1    .  .  .  .  .  .  .  .  .  .
        2    .  .  .  .  .  .  .  .  .  .
        3    .  .  .  .  .  .  .  .  .  .
        4    .  .  .  .  .  .  .  .  .  .
        5    .  .  .  .  .  .  .  .  .  .
WEST    6    .  .  .  .  .  .  .  .  .  .        EAST
        7    .  .  .  .  .  .  .  .  .  .
        8    .  .  .  .  .  .  .  .  .  .
        9    .  .  .  .  .  .  .  .  .  .
       10    .  .  .  .  .  .  .  .  .  .

                         SOUTH
```

Homebase is 1,1
(Call it a matrix if you wish)

BA

LOOKING OVER THE MEDIA - JRB

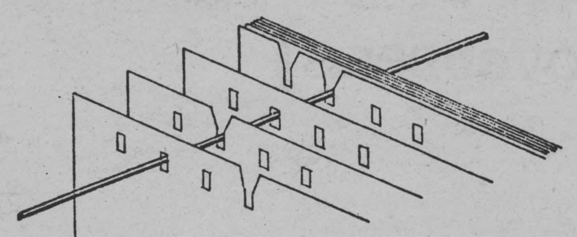
JERRY HEREBY EXPRESSES HIS ADMIRATION FOR THOSE WHO DRAMATICALLY EDUCATE US AT GREAT PERSONAL COST.

MISS SKINNER
TEACHER
MATH
GEOG.
READ
WRIT.
COND.
TARDY?
VOLTS
IMAG.
CHAR.
RIGHT
LEFT
IMPRINTING

Drawing by ARTHUR BERGER (Reprinted from WORKFORCE, Jan-Feb, 1973)

A WAY TO STORE & RETRIEVE INFORMATION

THIS IS A SIMPLE TECHNIQUE FOR STORING, CROSS REFERENCING AND RETRIEVING INFORMATION. IT IS BASED ON AN OLD, SIMPLE PRINCIPLE ILLUSTRATED IN THE FIGURE BELOW. IF THE WIRE IS LIFTED, THE SECOND & FOURTH CARDS WILL FALL OUT. IF THE SLOTS (▼) HAD BEEN CUT ONLY IN CARDS WITH RECIPES FOR BREAD WRITTEN ON THEM, THE OPERATION WOULD HAVE SEPARATED ALL OF THE BREAD RECIPES FROM THE REST. THE MAIN ADVANTAGE OF THIS SYSTEM IS THAT ONE CAN EASILY CROSS REFERENCE INFORMATION. FOR EXAMPLE, IF SLOTS HAD BEEN CUT ABOVE THE FIRST HOLE ONLY IN CARDS THAT HAD RECIPES WITH NUTS ON THEM, ONE MORE SORT WOULD HAVE RESULTED IN ALL OF THE NUT BREADS BEING SEPERATED. IT IS OBVIOUS THAT ANY TYPE OF INFORMATION THAT CAN BE WRITTEN ON CARDS CAN BE STORED & ACCESSED THROUGH THIS TECHNIQUE WITHOUT CONCERN FOR KEEPING THE CARDS IN ORDER.



THERE ARE SEVERAL PRODUCTS BASED ON THIS PRINCIPLE, BUT THEY ARE FAIRLY EXPENSIVE. IBM CARDS WORK JUST AS WELL & ARE READILY AVAILABLE & EASILY PUNCHED AT YOUR LOCAL COMPUTATION CENTER.

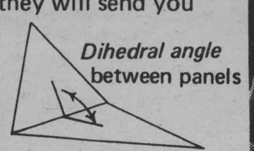
I HOPE YOU CAN THINK OF OTHER USES FOR THIS SYSTEM SUCH AS GAMES, OR STORING INFORMATION ON CIA ACTIVITIES IN CENTRAL AMERICA, ETC. I FEEL SO CLEVER EVERY TIME I STICK THE WIRE IN & ALL THOSE CARDS FALL OUT. MAYBE YOU WILL TOO.

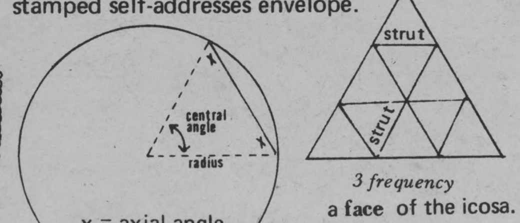
W. POTTS

from – de school primer no. 7
Zephyros Materials Exchange
1201 Stanyan St.
San Francisco CA 94117

THINKING OF BUILDING A DOME?

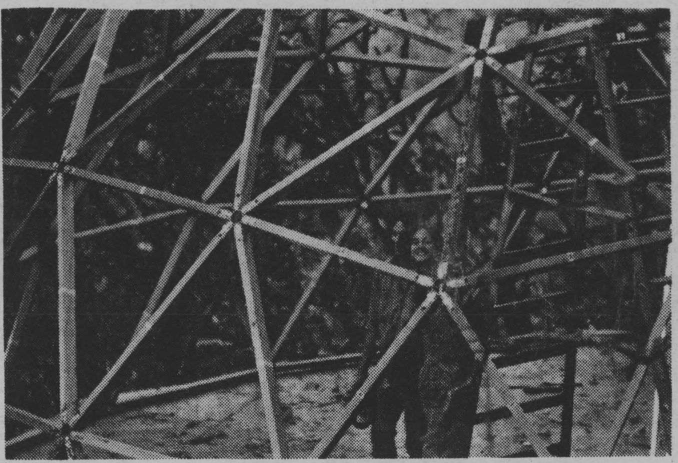
After you've read DOMEBOOK II ($4 from Shelter Publications, Box 279, Bolinas, CA. 94924), you can have a computer do or check your calculations. Send the frequency and actual radius of your proposed icosa-alternate breakdown (not triacon) dome to Resourse One, 1380 Howard, San Francisco, CA 94103, and they will send you
*lengths of struts
*the axial angles
*the central angles
*the dihedral angles
*the spherical angles
Be sure to enclose $1 and a stamped self-addresses envelope.





x = axial angle

3 frequency
a face of the icosa.



Joe and Robin the day they finished the frame for their dome. They built a two story house, the second story being a 30' diameter 4-frequency dome with a 3/8 sleeping loft inside. From Jerry Brown's DOMEFILM, a 16mm color film documentary on dome building.

MEDIA MIX

An interesting newsletter entitled *Media Mix: Ideas and Resources for Educational Change* recently shuffled its way to the surface of the mess on my desk. It carries interesting reviews on films, filmstrips, publications and other items. *Media Mix* is published monthly, October to May at 145 Brentwood Dr., Polatine, Ill.,60067 by Jeffery Schrank. One year subscription is $5, two year $9. Here is a sample item:

Lifestyle 2000: Inquiry into the Future . . . . Schools are only beginning to recognize that a study of the future is a valid part of the curriculum. In response to this recognition a few companies are producing media material about the future. One of the best to come along so far is the Denoyer-Geppert filmstrip Lifestyle 2000. The sound filmstrip is in four parts, each about 80-frames in 10 minutes. Each consists of an interview with a futurist--Hugh Downs, Paolo Soleri, Herman Kahn and Ray Bradbury. The four segments, however, are economically placed on two rolls of film. The Hugh Downs segment provides a gentle and general introduction to the study of the future paving the way for the mind-blowing ideas of Paolo Soleri. Soleri works mostly with young people in Arizona building his city of the future--a gigantic single building that serves as a total environment. Herman Kahn talks of knowledge and the ability to control the future and Ray Bradbury delivers a kind of pep talk encouraging optimism.

$30.60 from Denoyer-Geppert, 5235 Ravenswood Ave., Chicago, IL 60640. Also ask for their rapidly growing catalog of filmstrips.

COMPUTERWORLD
THE NEWSWEEKLY FOR THE COMPUTER COMMUNITY
SIXTY AUSTIN STREET, NEWTON, MASSACHUSETTS • 02160 • TELEPHONE: (617) 332-5606

Name That Hit Tune! But Computer Can't

HOLLYWOOD, Calif. – Computers can create music, but they cannot determine what makes a hit record, according to Allen D. Allen.

Allen, a composer and a research scientist, worked extensively with computers trying to find technical parallels among hit songs.

"Initially I tried to find a correlation between melodies I like," Allen said.

"I took all the tunes – the chord structures, tendency toward intervals, surprise, tempo, time signatures and all the other elements. There was absolutely no correlation between the songs I like. So I thought that maybe I'm weird.

"Then we did a study with an IBM 1130 computer. This was to find what hit records had in common. We used all the same factors (as for tunes) plus tone colors, instrumentations, lyrics, bottom or high end, etc. We could find absolutely no mathematical formula for a pleasing melody or a big-selling disk."

Games Humanoids Play
GUINDON

"Any people anywhere, being inclined and having the power, have the *right* to rise up, and shake off the existing government, and form a new one that suits them better. This is a most valuable, a most sacred right--a right, which we hope and believe, is to liberate the world."
--Abraham Lincoln

Wounded Knee Massacre (RERUN) 1973

send check or money order to: People's Computer Company
P.O. Box 310
Menlo Park, Ca 94025

name ____________________
address ____________________
____
____________________ zip

what kind of computer do you use? ____________________

✻ subscriptions start with 1st issue of school year

$4 for 5 issues
($5 Canada & overseas)

Bulk Rate
U.S. Postage
PAID
Menlo Park, CA.
Permit No. 371

TO:

Serial Supervisor, ERIC Clearinghouse
D 10 Cypress Hall
Stanford University
Stanford, Ca. 94305

PEOPLE'S COMPUTER CO.

APRIL, 1973
VOL. 1, NO. 4